CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

MEASUREMENT & CONTROL

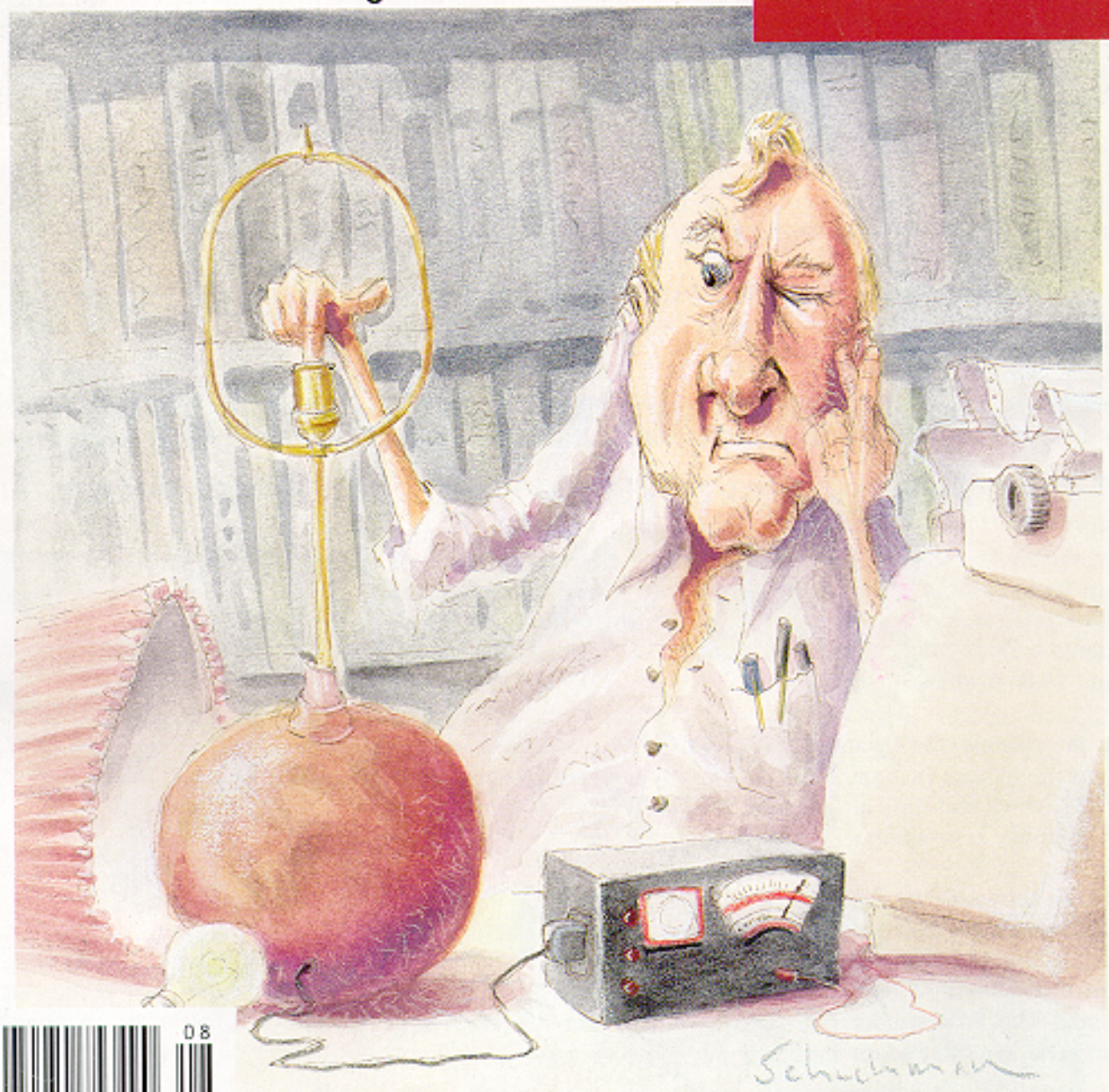
Introduction to Virtual Reality

Amplifier Design using Simulation

Battery Charger Supervisor Chips

Optical ID Card Reading

August 1993 — Issue #37

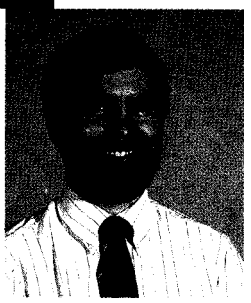


08
0 74470 75349 0

$3.95 U.S.
$4.95 Canada

EDITOR'S INK

Pavlov Would Be Proud



In some ways, this month's theme is rather redundant. While I'm not suggesting that "measurement" and "control" are synonymous, and you can certainly have measurement systems that aren't even remotely connected to control, I do assert that you can do only very limited control without some sort of measurement or real-world input to the system. Even on a factory assembly line, where dozens of operations are being performed over and over day in and day out, without some kind of feedback to the system, how would it know when something went wrong that needed fixing? Yes, there are some very stupid machines out there that require human supervision the entire time they are operating, but what good is automation when the tedious human element is still involved?

Along those lines, you usually need a good amount of parallel I/O for doing both measurement and control. The IBM PC's output-only printer port is pretty worthless for such a task, and the Macintosh SCSI interface is daunting to many designers. To correct both situations, we have a pair of articles this month that deal with basic interfacing issues related to both the PC and the Mac.

The PC Parallel Expander plugs into any standard (?) PC printer port and provides 16 inputs and 16 outputs (with a bit of coding voodoo thrown in to make the whole thing work). On the Mac side, Marc Bumble covers the basics of putting together a rudimentary Mac SCSI interface that can be expanded into any number of applications.

Another prime example of user input driving a response is the up-and-coming world of virtual reality. By definition, a VR system generates a display (and sometimes physical motion) based on a user's body movements. While the subject of VR can fill volumes, we get you started with a discussion of the basics of virtual reality and how you can get started with VR using your desktop PC.

On a much smaller scale, the idea of feedback affecting the final output almost always shows up in amplifier design. Our fourth feature article shows you how to use computer-based simulation to ensure your latest amplifier design is stable across its range of operation.

In the regular departments, Ed continues with the hardware enhancements to his embedded '386SX by adding a watchdog. Jeff starts a two-part series exploring an interesting cross between product bar codes and magnetically encoded credit cards: optical ID cards. Speaking of embedded PCs, Tom presents an overview of the present "embedded PC" marketplace and gives you plenty of resources to investigate. John concludes his pair of articles on battery supervision and charging by looking at some potent chips that take the burden off the designer. Finally, Russ takes a look at patent abstracts that relate in some way to making life for the handicapped a little easier.

Ken

CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcia

EDITOR-IN-CHIEF
Ken Davidson

TECHNICAL EDITOR
Michael Swartzendruber

ASSOCIATE EDITOR
Robert Rojas

ENGINEERING STAFF
Jeff Bachiochi & Ed Nisley

WEST COAST EDITOR
Tom Cantrell

CONTRIBUTING EDITORS
John Dybowski & Russ Reiss

NEW PRODUCTS EDITOR
Harv Weiner

ART DIRECTOR
Lisa Ferry

GRAPHIC ARTIST
Joseph Quinlan

CONTRIBUTORS:
Jon Elson
Tim McDonough
Frank Kuechmann
Pellervo Kaskinen

Cover Illustration by Bob Schuchman
PRINTED IN THE UNITED STATES

PUBLISHER
Daniel Rodrigues

PUBLISHER'S ASSISTANT
Susan McGill

CIRCULATION COORDINATOR
Rose Mansella

CIRCULATION ASSISTANT
Barbara Maleski

CIRCULATION CONSULTANT
Gregory Spitzfaden

BUSINESS MANAGER
Jeannette Walters

ADVERTISING COORDINATOR
Dan Gorsky

CIRCUIT CELLAR INK. THE COMPUTER APPLICATIONS JOURNAL (ISSN 0896-8985) is published monthly by Circuit Cellar Incorporated, 4 Park Street, Suite 20, Vernon, CT 06066 (203) 875-2751. Second classpostagepaidat Vernon, CT and additional offices. One-year (12 issues) subscription rate U.S.A. and possessions $21.95, Canada/Mexico $31.95, all other countries $49 95. All subscription orders payable in U.S. funds only, via international postal money order or check drawn on U S. bank. Direct subscription orders and subscription related questions to The Computer Applications Journal Subscriptions, P.O. Box 7694, Riverton, NJ 08077 or call (609) 786-0409
POSTMASTER: Please send address changes to The Computer Applications Journal, Circulation Dept P 0. Box 7694, Riverton, NJ 08077.

Circuit Cellar BBS—24 Hrs. 300/1200/2400/9600/14.4k bps.8 bits, no parity, 1 stop bit, (203) 871-1988; 2400/9600 bps Courier HST, (203) 871.0549

INSIDE ISSUE 37

NEW PRODUCT NEWS

Edited by Harv Weiner

TONE DECODER WITH SERIAL PORT

International MicroPower has introduced a Commercial Telephone Line Decoder that combines many features at a low price. The Digit Snatcher II simplifies the capture and storage of digital tones by means of an LCD display with built-in help menus. An Intel microprocessor controls the storage of thousands of digits, offers automatic help messages, and sends and receives serial RS-232 data.

The Digit Snatcher II also incorporates Caller ID capture. A built-in microphone with electronic automatic level control and noise filter allows acoustic capturing of DTMF dialing as well as Caller ID, eliminating the need for an electrical connection between the source and Digit Snatcher II. The unit will decode and store DTMF signals from acoustic signals coming from TV or radio as they are heard.

The unit features a storage capacity of up to 4000 digits. A built-in day, date, and time stamp option is available, marking each series of digits with the current date and time. Stored information will be retained for up to 5 years, even while the unit is turned off, which means the Digit Snatcher II can be taken into the field to decode and store digits, and later be connected to a desktop or laptop computer for data retrieval.

A 5-mm coaxial DC power jack is standard, but the unit will work for up to 26 hours on an internal 9-V battery. A "one-button" locking device allows the entire unit to be opened for battery access in less than 5 seconds. The compact hand-held unit comes in a hard anodized extruded aluminum case, which makes it resistant to scratches and marks. It can be easily cleaned with a damp cloth.

The clock/calendar option is easy to use and continues to keep track of the date and time while the unit is off. Setting the date and time is accomplished in the same manner as a simple digital clock and automatic correction for short months and leap years is included.

The Digit Snatcher II features help menus for ease of operation. The unit will prompt with choices if an appropriate selection is not entered.

The Digit Snatcher II sells for $179 with 1000 digits of storage. A 2000 digit storage unit with Caller ID and serial port sells for $289. All options sell for $550. A 20-page operator's manual is available on request.

International **MicroPower** Corp.
65 Palm Dr. • Camarillo, CA 93010
(805) 482-2870 . Fax: (805) 389-1274 **#500**

SOLID-STATE TEMPERATURE MEASURING DEVICE

A solid-state, user-modifiable temperature sensing device that requires no batteries has been introduced by Parham P. Baker and Associates Inc. The Temp-A-Chip interfaces to any RS-232 serial port and enables temperature monitoring from the computer.

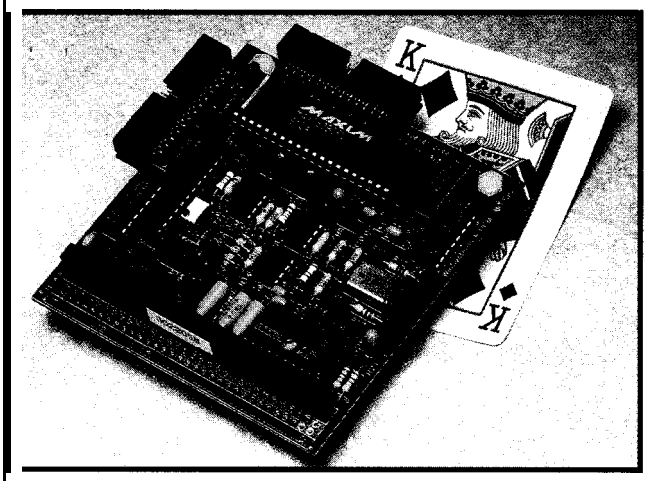
Unlike other temperature sensors, the Temp-A-Chip provides a more linear measurement of temperature because of its solid-state design. No batteries are required, and the Temp-A-Chip software package can be modified to meet specific needs.

The Temp-A-Chip is fully powered from a 9600-bps standard serial port (XT or AT connector available) and is useful over a temperature range of O-l 15°F (-1746°C). It features an LCD screen with constant temperature readout. The unit is programmable from Windows or DOS and may be controlled from any communications package.

The Temp-A-Chip sells for $99.95 plus shipping and handling. A 30-day money back guarantee is provided.

Parham P. Baker **&** Associates, Inc.
153 Burt Rd. . Lexington, KY 40503
(606) 278-8699 . Fax: (606) 277-7514 **#501**

NEW PRODUCT NEWS



LOW-COST ANALOG I/O MODULE

A complete 12-bit analog input/output module for PC/104-compatible embedded systems has been introduced by WinSystems. The **PCM-AI0** provides affordable, high-speed data acquisition and control functions with conversion speeds of 10 microseconds per channel.

The heart of the board is the Maxim MAX180 12-bit data acquisition chip. This device combines an 8-channel input multiplexer, high-bandwidth track-and-hold, a low-drift zener reference, high-speed successive-approximation analog-to-digital converter (ADC), and flexible microprocessor interface on a single chip. It supports up to eight single-ended or four differential analog inputs which are software selectable on a per-channel basis. The MAX180 samples and digitizes at a 100-kHz throughput rate.

The PCM-AI0 also contains an Analog Devices AD7537 dual 12-bit digital-to-analog converter (DAC). Two idependent DACs are in one monolithic chip that is configured to provide two 0 to ±5-volt outputs. The input ection is double buffered to allow simultaneous update of both DACs. These registers latch the 12-bit digital word ıd keep the D/A converter's output constant until it is updated with a new value in one step.

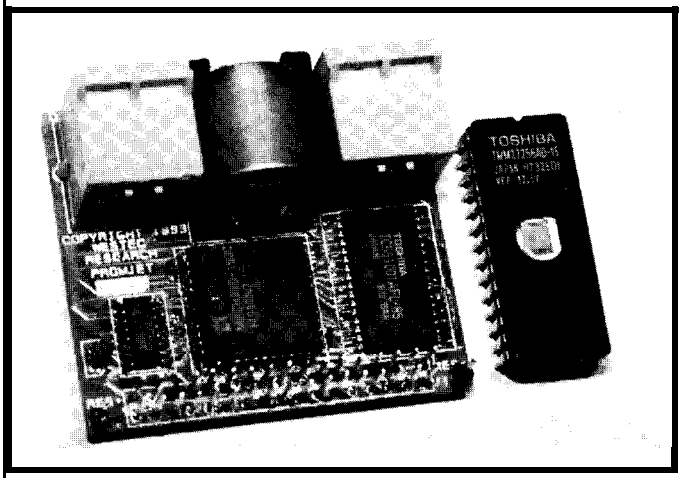
The PCM-AI0 operates over the temperature range of -25 to +85°C. The module contains low-power CMOS ɔgic devices to reduce current draw and increase product reliability. It requires only 200 milliwatts of power. The nit measures only 3.6" by 3.8". It is an 8-bit stackthrough module that can be used in a stand-alone stack or as a ıezzanine bus stacked atop a larger single-board computer.

The PCM-AIO sells for $295 and carries a two-year warranty. The PCM-AIO-80, a lower-cost version offering channels of A/D input only, sells for $250.

WinSystems, Inc.
15 Stadium Dr. . Arlington, TX 76011 . (817) 274-7553 . Fax: (817) 548-1358 **#502**

COMPACT EPROM EMULATOR

An ultracompact .PROM emulator from Vestec Research mulates all EPROMs from 64K (8K x 8) to 8M (1M x 8). The **PROMJet** is contained on a 2.2" x 1.9" PC board and features battery-backed high-speed RAM, a download rate of 1 Mb/s, and easy-to-use software.



The PROMJet connects to the EPROM socket of the system under development and the printer port of a PC. After downloading the data from the PC, the PROMJet resets the target system and emulates its EPROM. The PROMJet is software configurable (no jumpers) and operates in both DOS and Windows environments.

Multiple PROMJets allow 16-, 32-, 64-, and 128-bit emulations. Options include a 40-pin DIP adapter, 32- and 44-pin PLCC adapters and 40-ns emulation.

The PROMJet sells for $295 in a 2M (256K×8), 85-ns version. A 4M (5 12K x 8) sells for $495 and an 8M (1M×8) sells for $695.

WesTec Research Corporation
2750 Riverside Dr., Ste. 205
Los Angeles, CA 90039
(213) 664-8909

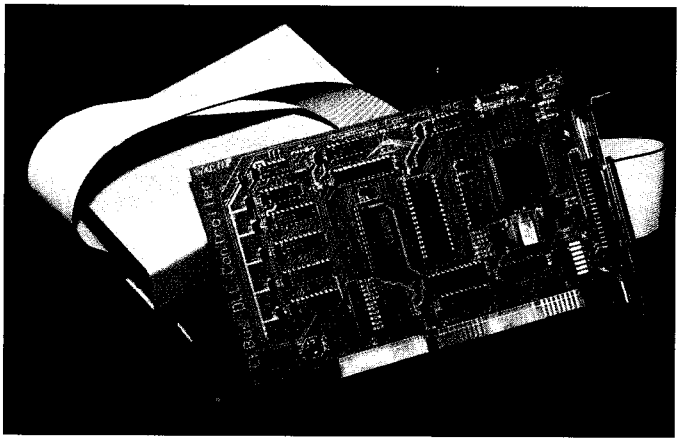
#503

NEW PRODUCT NEWS

CEBUS PROTOCOL ANALYZER

CEBugger, a CEBus protocol analyzer from Command Control Inc., provides the developer with an easy way to observe and analyze a CEBus network. It allows the capture, display, and analysis of CEBus packets. CEBugger may be set to filter the packets or trigger a capture on a specific packet or event. CEBugger will check for errors and protocol violations.

The CEBugger package consists of a 16-bit IBM AT-bus card, a CEBus modem, and software that runs on the PC. A 16-MHz 80C196KC microcontroller on the card executes the CEBus Data Link Layer (DLL) software. This software is loaded onto the card (through the PC's DMA channel) at runtime, so the same card may be used with CEBugger, CEBnode, or other programs without changing EPROMs.



Updates for both CEBugger and the DLL software are available from an on-line BBS for registered users.

CEBugger incorporates multilevel error checking and identifies four different classes of errors: media errors, such as loss of carrier, bad checksum, and noise bursts; notifications (nonstandard NPDU or DLL control field); warnings (borderline timing errors); and protocol violations. Error checking for each of these classes may be independently enabled or disabled.

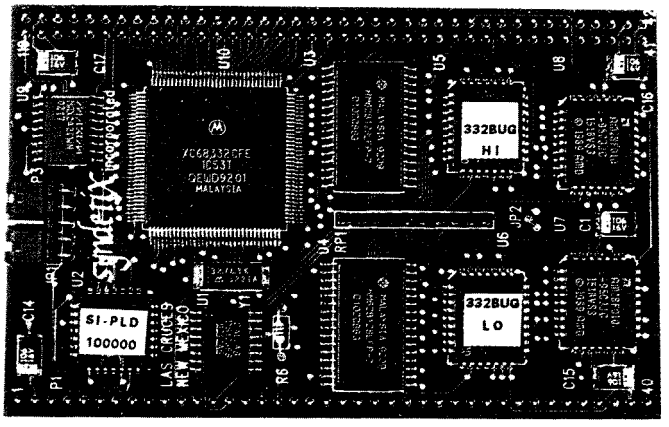
The CEBugger Protocol Analyzer for power line sells for $3095. Analyzers for twisted pair, infrared, and coax are available for $2995 each.

Command Control, Inc.
8800 **Roswell** Rd.,
Ste. 130
Atlanta, GA 30350-1875
(404) 992-8430
Fax: (404) **993-3603**

#504

EMBEDDED CONTROLLER

The **Syndetix Embedded Controller** (S.E.C.) is designed for systems that require powerful controller functions. With its zero-wait-state Flash memory and low power consumption, it is ideal for in-circuit programmable embedded controller applications.



The small (4.11" x 2.61" x 0.4") board features an MC68332 or MC68331 CPU, 256K or 1 MB of SRAM, 256K or 512K Flash memory, 128K EPROM, and a built-in RS-232 interface. Power requirements are only 180 mA at 5 volts and 16.67 MHz. Sleep functions are included to conserve power. The on-board EPROM contains Motorola CPU32Bug with additional commands for loading the Flash memory directly from the serial port. The combination of on-board CPU32Bug and Flash memory speeds development and adds greater flexibility when software modifications are required.

The S.E.C. is suitable for data acquisition, process control, and other real-time applications. Software is developed and loaded directly into the on-board Flash memory. After the software has been fully tested, a removable jumper allows the CPU to boot directly to the application code. The SRAM may be externally battery backed, and the RS-232 port may be turned on and off as required with external circuitry to conserve power.

The S.E.C. sells for $750 in single quantity. The price includes a comprehensive user's manual as well as Motorola manuals on the CPU and CPU32Bug.

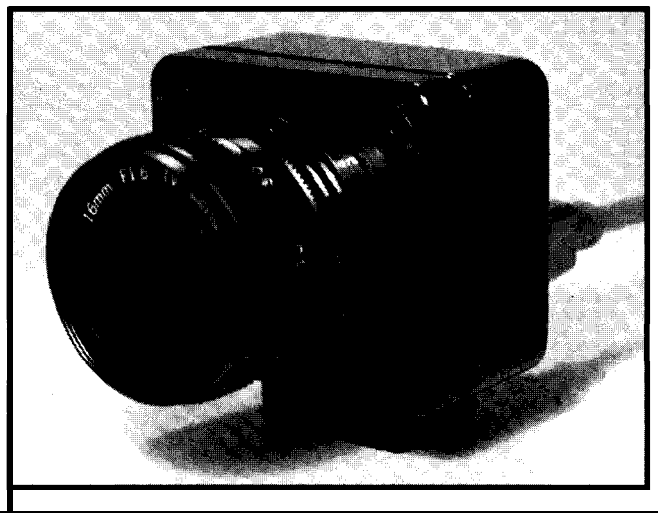
Syndetix, Inc.
2820 North Telshor Blvd.
Las Curses, NM 88001
(505) 522-8762
Fax: (505) 521-1619

#505

NEW PRODUCT NEWS

ELECTRONIC COLOR IMAGER

Digitized 24-bit color images with a resolution of 75 1 x 488 pixels can be accomplished with a new high-resolution color camera from Electrim Corp. Applications for the device include desktop publishing, machine vision, document imaging, security, industrial inspection, and telecommunications.



The **EDC-1000C** color imager interfaces directly to an IBM PC/AT or compatible and digitizes images into 8 bits each of red, green, and blue for storage in the PC's RAM. The camera uses a frame transfer CCD image sensor to provide a resolution of 75 1 x 488 interlaced or 75 1 x 244 noninterlaced.

Notable features of the camera include no dead space between pixels, computer-controlled exposure time, and data collection rates up to 1.6 MB/second (3 to 5 frames/second in live mode). TIFF, PCX, and Targa file formats are supported.

The camera can be used with virtually any Super VGA card that supports VESA (Video Electronics Standards Association] BIOS extensions version 1.2, and resolutions of 800x600 or 640x480 with 16-bit color.

The EDC-1000C camera and software sell for $950. The EDC-1000HR camera (751 x 488 pixels) sells for $850 and the EDC-1000 camera (192 x 330 pixels) sells for **$400**.

Electrim Corp. • Electronic Imaging
P.O. Box **2074** • Princeton, NJ 08543
(609) 683-5546 • Fax: (609) 683-5882 **#506**

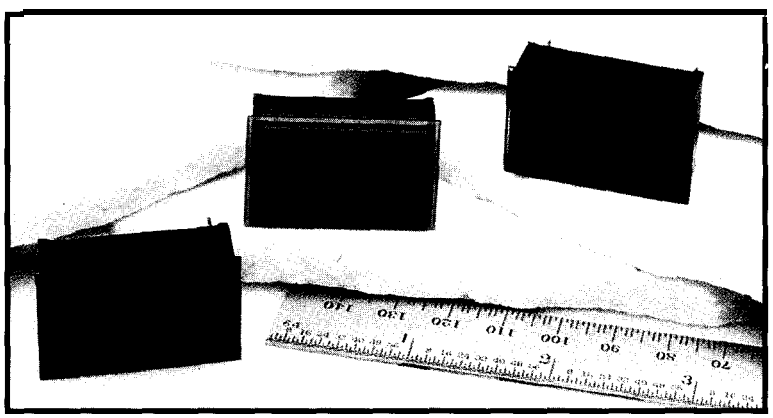
NEW PRODUCT NEWS

SUBMINIATURE DIGITAL VOLTMETER

A fully functional $3\frac{1}{2}$-digit precision digital voltmeter occupying just over a half cubic inch total volume has been announced by Date1 Inc. These self-contained, plug-in modules provide research-grade accuracy, reliability, and low cost in a component-size DDIP package.

The **DMS-20PC** is available in signal input configurations ranging from ±0.2 V to ±200 V. The display can be in several colors including high-density red and low-power red (less than 7-mA power drain). The units feature a large (0.37") LED display, have an integrated bezel, and are fully encapsulated to withstand harsh environments. All models feature high-impedance (typically 1000 MΩ) differential inputs, autozero display, and autopolarity indication while employing an ultrastable reference circuit, Decimal point placement is user selectable.

Long-term stability is achieved through an advanced autozeroing ADC which never requires adjustment or calibration. Typical accuracy ranges from ±1 count to ±2 counts. All meters are overvoltage protected to ±250 V with common mode voltage range of ±2.0 V. An optional HOLD/RUN pin may be ordered, if desired. The display enable option allows the meter to be powered down when not in use. The DMS-20PC starts at $29 each.



Datel, Inc.
11 Cabot Boulevard • **Mansfield, MA** 02048
(508) 339-3000 • Fax: (508) 339-6356

#507

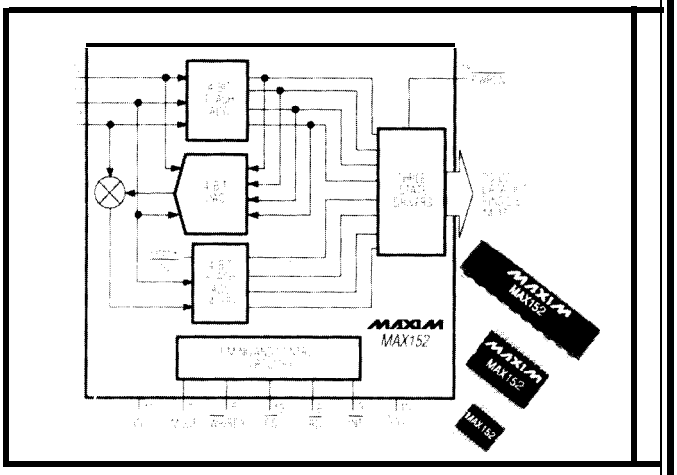
NEW PRODUCT NEWS

MICROPOWER A/D CONVERTER

A micropower A/D converter that provides full S-bit performance with a 3-volt supply has been introduced by Maxim Integrated Products. The **MAX152** uses a half-flash conversion technique to achieve a 1 .8-µs conversion time and digitizes at a rate of 400k samples per second. A power-down feature extends battery life at reduced sampling rates by cutting the supply current to microamp levels. The 20-pin SSOP package occupies 30% less area than an S-pin DIP.

To minimize battery drain during burst-mode conversions, the converter powers down quickly and then powers up again within one conversion period. Supply current drops from 1.5 mA (3 mA maximum) to 1 µA following a power-down command. The device powers up in less than 1 microsecond maximum, including 450 µs for signal acquisition by the internal track/hold circuit.

The dynamic specifications for the MAX152 include 45 dB minimum SiNAD and -50 dB maximum Total Harmonic Distortion (THD). Its microprocessor interface appears as a memory location or I/O port and requires no external interface logic. The data outputs use latched three-state buffered circuitry for direct connection to a microprocessor data bus or system input port. Vin and Vref terminals allow ratiometric operation.



The MAX152 sells for $4.25 in quantity.

Maxim Integrated Products
120 San Gabriel Dr.
Sunnyvale, CA 94086
(408) **737-7600** **#508**

FEATURE ARTICLE

Joseph D. Gradecki

An Introduction to PC-based Virtual Reality

Virtual Reality is one of today's hottest development areas. As usual, the mass media (movies especially) have latched on to the concept and blown it out of proportion. Find out what low-cost VR really is.

Since the release of the motion picture "The Lawnmower Man," everyone has become obsessed with the technology of Virtual Reality (VR). While VR is just making its way into the mainstream, it has been around for many years. In this article, I'll explore the topic of Virtual Reality using an IBM-compatible personal computer.

WHAT IS IT?

Many definitions have been given for Virtual Reality by press and industry figures. However, I feel the most accurate definition for VR is "an interactive three-dimensional playground." Using a computer attached to some specialized hardware that's running some clever software, a VR user is put into a virtual "world" built from the developer's imagination. The software represents the visual aspects of the virtual world as a number of shaded polygons that may or may not have visual textures or other attributes.

In the most rudimentary systems, the user wears a pair of shutter glasses which block one of the eyes at the same time an image is flashed on the screen. The glasses cause the images on the monitor to appear three-dimensional. The user can upgrade to

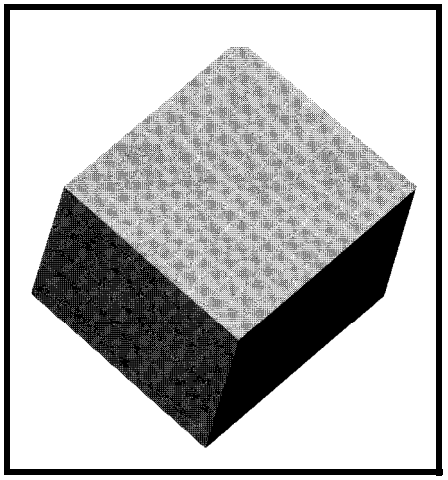


Figure 1—*Typical renderers take in raw numeric data and create solid objects with proper* ***perspective to give*** *the illusion of fhree dimensions.*

head-mounted display hardware to enhance the illusion of three-dimensional objects. A head-mounted display has two Liquid Crystal Displays (LCD)-one in front of each eye—which display slightly separate images. The brain fuses the images into a three-dimensional world.

Additionally, the user might use some kind of three-dimensional input device like a glove wired with sensors to interact with the virtual world. This interaction is what separates a three-dimensional game and a VR application. This does not come cheap. Current pricing for "top of the line" VR systems can range from $80,000 to $500,000 depending on the system's capabilities and the user needs.

CONVERTING YOUR IBM PC TO A VR MACHINE

An alternative to the high-priced systems is a homebrew setup. Using several simple interface circuits, a developer can add the Mattel Powerglove and Shutter Glasses from Sega or Toshiba to the parallel and/or serial ports of an IBM-compatible PC. These two pieces of hardware enable the user to interact in a three-dimensional virtual world right in their home. As the user's interests advance, peripherals such as 3-D sound, head position tracking, and head-mounted display systems can be built and added to the system to give a more realistic sense of immersion in their virtual world. However, all the hardware is useless without software to control it.

VIRTUAL REALITY SOFTWARE

Software for a VR system is called a ***rendering package.*** This software takes numeric data and converts it into a picture such as the one shown in Figure 1. Using a variety of different object formats and files, very creative worlds can be designed for a user with any text editing program that can generate ASCII output.

The rendering software must also drive the shutter glasses and the other input devices. In the case of an input device, the software must allow the user to interact with the virtual world in a realistic fashion. The user should be able to pick up objects and rearrange them in real time. This interaction gives the user a sense of immersion in the world.

THE RENDERER

The basic functionality of a renderer is the same for low-cost renderers and high-cost renderers. Figure 2 shows the loop that a simple renderer performs. In this section, I give a brief idea about what each of these steps entails.

```
Loop
  Get User
  Transform and Project Vertices
  Sort Objects
  Backface Removal
  Color
  Draw
Endloop
```

Figure *2--Renderers continuously repeaf the same basic set of steps in real time to create fheir illusions.*

GET USER INPUT

During user input, the computer program must provide a visual or auditory feedback to any number of user-generated inputs. The user could provide input to the computer through a keyboard or some other device. Typically, some sort of three-dimensional input device is preferred. The computer program must determine how much movement has occurred since the last interaction with any input device being used.

TRANSFORMS AND PROJECT VERTICES

When objects for a virtual world are described, they are put into ***world coordinates.*** World coordinates are based on a three-dimensional coordinate system. The projection of the coordinates of an object's vertices onto the computer screen coordinates requires several steps.

The first step in the projection of coordinate points between different coordinate systems is to convert the vertices from world coordinates to view space coordinates. The most common system for the view space is the ***perspective coordinate system.*** Figure 3 shows what a perspective view does to a cube drawn on the screen and the values used to create it.

The perspective view is used to create the illusion of depth in the screen image. The following formulas convert world coordinates to perspective view coordinates:

```
Vx = x/z * SCREEN_WIDTH/2
Vy = y/z * SCREEN_HEIGHT/2
```

Notice that the z coordinate stays the same from world to view coordinates. The last step in the projection is to convert the view coordinates to ***screen space coordinates.*** These coordinates are the actual (x,y) position of pixels on the screen that will make up the objects. Since there is no z coordinate for computer screens, it is simply discarded.

In addition to the projection of the object vertices, the computer program must move objects in accordance with the user's interactions with the input device. If the user wants a specific object moved some distance in the x coordinate direction, the computer program must recalculate each vertice's coordinate to adjust the vertices of the object accordingly. This adjustment is usually performed using transformation matrices. Below is an example of a transformation matrix for object translation (movement).

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

All vertices of an object have to be transformed using matrix multiplication. These calculations are obviously

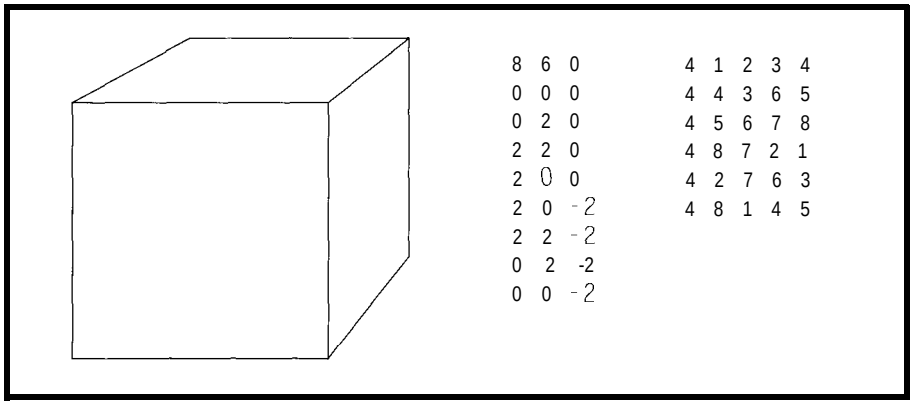


Figure 3—*The first step in the projection of coordinate points between different coordinate systems is to convert the vertices from world coordinates to view space coordinates.*

very compute intensive, because of the number of pixels involved, especially when considering that the renderer must work in real-time.

SORT OBJECTS ON Z DEPTH

Once all of the objects have been given view and screen coordinates, we sort the objects based on their z coordinate. The purpose of sorting is to determine which objects are in front of other objects. If we have two objects (A and B) and object A is in front of object B, the program will have to draw object B first and then object A to give the illusion of spatial, or depth, relationships between objects in the virtual world. The result of this is shown in Figure 4a. If the program were to draw A and then B, we would get the reverse as shown in Figure 4b. By sorting all the objects according to their *z* depth, we can always draw from the back of the list forward. In practice, the list is kept sorted at all times. When an object is transformed using a translation or rotation matrix, the object is located in the list and repositioned in the view space according to its new z coordinate.

BACK FACE REMOVAL

Back face or *hidden surface* removal is performed to save rendering time. If we have a cube in our world and we are looking at one of its sides, there is no need to render the opposite side of the cube since it will not be seen. Back face removal is a simple matter of determining the direction of the vector normal to a particular polygon's surface points. If the normal vector has a direction toward the user (the normal is greater than zero), the surface must be rendered. If the normal has a direction away from the user, the surface can be eliminated.

COLOR

Color is very important for adding another dimension of realism in the virtual world. Most renderers have the ability to specify point light sources in the virtual world. Each light source will have a direction and a color associated with it. As the renderer begins to draw a new screen, it will determine how much each of the light sources affects a certain polygon's surface color based upon the angle between the light and the polygon surface. If the polygon is directly in front of the light, then the full intensity of the light source is reflected from the polygon and it is colored accordingly. If the polygon is at an angle to the light source, then only the fraction of the light rays whose angle of reflection generates a ray which pierces the plane of the view space will be used to color the surface. By using a shading scheme, each of the polygon surfaces can have different shades of the same color based upon the intensity of the reflected light rays.

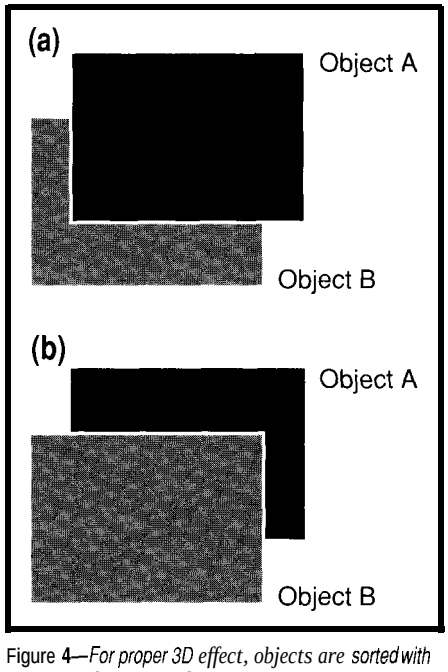


Figure 4—*For proper 3D effect, objects are sorted with respect to their z coordinate. (a) When object A is in front of object B, object B is drawn first. (b) Similarly, when object B is in front, object A is drawn first.*

DRAW

The last step in the rendering process is drawing the objects to screen memory. Significant time and energy is given to this subject by developers of rendering packages because of the amount of time spent drawing to the computer screen. The faster the line drawing routines, the faster the renderer can update the screen after some user input. The majority of this code can be written in highly optimized assembly language to take advantage of specific hardware. However, this limits the portability of the code, which serves to keep the prices of rendering packages high.

PROGRAMMING A VIRTUAL WORLD

In this section, I use the PCVR Renderer, (a rendering program that is being developed and described in PCVR magazine) to develop a Virtual World that consists of a grove of trees. The first step in creating a new virtual world is to draw the proposed world from an overhead two-dimensional view. This view gives me an idea of the scale I want to use when placing the trees. The next step is to place the objects in the world using the standard three-dimensional coordinate system. Using these preliminary setup steps allows me to see where the objects will be in the new world and the distances between them.

After I have placed the objects, I have to design each one of the objects. There are several different ways to develop objects:

*Create object "by hand"

*Create the object using Computer Aided Design software

•Use a public domain object

The first option, create by hand, relies on your ability to do three-dimensional art on a two-dimensional drawing pad. This option is good for very simple objects that contain boxes, triangles, and other rudimentary shapes. The second option works well when the object is quite complex and real three-dimensional views of the object are needed in order to perfect it. The last option is the most attractive because there is no sense in reinventing the wheel when somebody else has already done it. There are many objects already in the public domain that can be used to create a virtual world using the renderer.

For my example, I am going to use a public domain object and explain its features and how it was created. Figure 5 shows the printout of my tree object. After any optional header information comes the actual points or vertices used in the creation of the objects. These vertices are based in the three-dimensional coordinate system and are separated by spaces.

The vertices are followed by information about the polygons that make up the object. As stated earlier, the renderer uses polygons to represent objects just as they are defined in the object files themselves. Polygons can have from three to *n* vertices. For the object file, each of the polygons must be defined from the vertex list defined at the beginning of the file. The polygon definitions each begin with the color of the polygon to be defined. This number is followed by the total number of vertices that make up the polygon. Next comes the index number of each of the vertices in the polygon. The vertices are listed in 0 to *n*–1 order.

This description of the tree object file is specific to the PLG format. PLG is the data format for the public domain R E N D3 8 6 Virtual Reality renderer. There are many object file formats used throughout the world. The PCVR Renderer can convert from the majority of these formats.

The next step is to build the virtual world.

CREATING THE WORLD

Creating a virtual world is a simple matter of determining what objects you want in the world. Will *you* have trees and a park bench or just trees? After the objects have been placed in the world, you must determine from what direction the user will look into the virtual world. This is called the *viewpoint.* Viewpoints can

```
tree 26 25
          0 100 10 #0
          0  0  10 #1
          9100  5 #2
          9  0  5 #3
          9100  -5 #4
          9  0 -5 #5
          0100  -10 #6
          0  0 -10 #7
-9 100 -5 #8
-9  0  -5 #9
-9100  5 #10
-9  0  5 #11

28 100 -30 #12
-28 100 -30 #13
28 150 -60 #14
-28150  -60 #15
28 200 -30 #16
-28 200 -30 #17
28200  30 #18
-28 200 30 #19
28 150 60 #20
-28 150 60 #21
28 100 30 #22
-28 100 30 #23

65150  0 #24 point
-65 150 0 #25 point

0x17FF 4 14 15 13 12  #rect.sides
0x17FF 4 16 17 15 14
0x17FF 4 18 19 17 16
0x17FF 4 20 21 19 18
0x17FF 4 22 23 21 20
0x17FF 4 24 25 23 22

0x17FF 3 24 14 12     #pointy ends
0x17FF 3 24 16 14
0x17FF 3 24 18 16
0x17FF 3 24 20 18
0x17FF 3 24 22 20
0x17FF 3 24 12 22

0x17FF 3 25 23 13
0x17FF 3 25 21 23
0x17FF 3 25 19 21
0x17FF 3 25 17 19
0x17FF 3 25 15 17
0x17FF 3 25 13 15

0x12AA 4 2 3 1 0      #sides of trunk
0x12AA 4 4 5 3 2
0x12AA 4 6 7 5 4
0x12AA 4 8 9 7 6
0x12AA 4 101198
0x12AA 4 011110

0x17FF 6 1 3 5 7 9 11  # bot of trunk
```

Figure5—*Public domain objects, such as a tree, are plentiful and often save you from reinventing the wheel.*

be anywhere in the three-dimensional coordinate system. Are you going to be under the park or above it?

The last consideration is the presentation of the images. Is any special hardware being used? If so, you may choose a stereoscopic presentation. In the next three sections, I will address each of these areas.

OBJECTS AND JOINTS

The PCVR Renderer includes the ability to create any object such as the tree discussed earlier. The renderer itself includes a format that allows very precise handling of objects that can be confusing for beginning programmers. Therefore, I recommend building objects using the OFF format. This format allows for the creation of objects that can be used in a variety of other software packages and is freely transferable in public domain. The format is defined by the creation of two files called the *geometry file* (.geom) and the *header file* (.aoff). The header file includes the information shown in Figure 6.

The information in the property list is standard except for the color of the object, which is described in the common red, green, blue format. A value of 1 .O is full color intensity.

The geometry file is where the actual polygon is defined. It is essentially the same as the PLG file described above except the color information is in the header file. Figure 3b shows an example of a geometry file for a simple cube.

Once an object has been defined in the OFF format, it is converted to the PCVR Renderer using a conversion program called LOADOFF.EXE.

Once all of the object files have been created, the rendering package has the ability to create joints between them. The classic example of a series of joints is the human hand.

The developer of a virtual world wants to see a hand in a program so the user can grab things. In order to model the hand correctly, the developer creates a palm object and objects for each of the finger and thumb segments. Using the JOINTS file, the developer creates joints between the palm and the first segment in each of

```
name
description
author copyright type usually POLYGON

# Property list for this object
ii Prop.              data type      format   filename or default data
ii-------------------------------------------------------------------
    geometry          indexed_poly   fff      filename.geom
    vertex-order      default        s        clockwise
    polygon_colors    default        fff      1.0 1.0 1.0
    back-faces        default                 s cull
```

Figure 6—*The popular OFF format uses a pair of files to create an object and include the geometry file and the header file (shown above).*

the fingers and the thumb. The developer further creates joints for each of the segments in the hand.

Joints not only connect objects but allow the developer to limit the movement of each of the objects based on the movement of jointed objects. Thus, if the palm of the hand moves to the left in the world, the finger will follow because they are jointed. If any of the finger segments is rotated, the jointed object rotates as well. If a limit is placed on the rotation of one of the objects, it will not rotate beyond this limit even if a jointed segment is rotated further. Limits can also be imposed on the placement in the world, such as limiting the forward motion of the object.

To illustrate the format of a JOINTS file, we will look at placing two cube objects in a world and creating a joint between them. I should note that objects do not have to be touching to be jointed.

All JOINTS files have a root object. A pointer to this object is returned when the read_joint function is called. The read_joint function accepts a filename string as a

```
typedef  struct_viewpoint{
        Xform view-matrix;
        TAngle pan,
              tilt,
              roll:
        int   x,
              y,
              z;
        Fixedpoint stereo-d,
                   stereo-e;
      IVIEWPOINT:
```

Figure 8—*The VIEWPOINT structure contains all the information about the location of the user relative to the scene being observed.*

parameter and returns the root object after reading the joint file successfully. The word root is followed by a virtual word for the root object. Instead of using obscure filenames for the name of objects in the joint file, virtual words are used. For this example, I will call the first cube object *cube 1*. To set the root, I use the following:

```
ROOT cube1
```

The next part of the joint file defines all of the objects that will be used in the joint file. I will use two cubes and place them in different locations of the screen. The first cube is defined as:

```
name  cube1  cube.obt
translation  0  0  -950
```

The keyword name indicates that a new object is being defined. This is followed by the virtual word for this object, which in turn is followed by the filename for the object. The translation keyword tells the renderer to place the first cube at the coordinate position (O,O,-950). The second cube is defined as:

```
name  cube2  cube.obt
translation  100  0  100
```

The last line for this joint file actually creates the joint:

```
joint  cube1  cube2
```

This line creates a joint between the objects cube1 and cube2. The object cube 2 is a descendant of the object cube 1. Thus, any movements or rotations performed on cube 1 will affect cube 2, but movements on cube 2 will not affect cube 1. Joints work on a tree concept, where actions fall down the tree but not up.

YOUR VIEWPOINT

The position in which you view a virtual world makes a difference. One of the exciting things about virtual reality is the ability to view a world from any viewpoint. You can get

```
VIEWPOINT *view_one;

if((view_one = create-viewpoint (0.0,0.0,300.0,0,0,0))==NULL)
   { printf ( "View creation failed./n"); exit(1); }
```

Figure 9—*At initialization time, a viewpoint structure is set up at a default coordinate of (0,0,0).*

inside an object and view the world from the object's viewpoint. You can fly like a bird and see what it sees. In the PCVR Renderer, your viewpoint can be anything you want simply by specifying a three-dimensional coordinate. A VR program sets its viewpoint with the function `create_viewpoint`. Thisfunction returns a pointer to a structure of type VIEWPOINT. Figure 7shows this structure. An example of a complete viewpoint setup is given in Figure 8.

The new viewpoint is located at the origin in the world, or (O,O,O). We have the ability to move the viewpoint to any location at any time.

Each time the viewpoint is changed, the renderer recomputes the position of the objects in the world and redraws the screen. One of the most powerful features of VR software is the ability to define several different viewpoints. By defining several viewpoints, the user can instantly change the direction they are looking just by pressing a key on the keyboard or by using some other input device. For instance, imagine being in a room and wondering who is knocking on the door. Instead of opening the door, you simply change viewpoints to outside the room to see who is knocking.

ONE OR TWO EYES

Finally, when a user is using just the computer screen to view a virtual world, they see a single image of the screen. This is called *monoscopic presentation.* The renderer draws a single image of the objects in the world on the computer screen and the user relies on human ability to bring out the depth in the image. The developer of this world helps to facilitate the depth by using the perspective view technique and making farther objects smaller than objects that are closer to the user.

To better achieve the true sense of three dimensions, a user can wear shutter glasses or a head-mounted display. When these pieces of equipment are used, the rendering software must generate two separate views of the world. One of the views is for the left eye and the other is for the right eye. This is achieved by moving the viewpoint of the user a little to the left and generating an image, then moving the viewpoint a little to the right and generating an image. Depending on the hardware used, each of the images is presented to the appropriate eye and the user sees a true 3-D image.

CONCLUSION

In this article, I touched on the hardware and software necessary to bring Virtual Reality to the IBM-compatible personal computer user. The renderer provides the capability necessary for the creation of sophisticated virtual worlds and the interactions in these worlds. ❑

In addition to being the publisher of PCVR magazine and the Director of Software Development at VRontier Worlds of Stoughton, Inc., Joseph holds a Bachelor's degree in Computer Science and is currently working on his Master's degree in Computer *Science.*

SOFTWARE

Software for this article is available from the Circuit Cellar BBS and on Software On Disk for this issue. Please see the end of "ConnecTime" in this issue for downloading and ordering information.

CONTACT

Those interested in more information about Homebrew and Low End Virtual Reality Technology are directed to:

PCVR
P.O. Box 475
Stoughton, WI 53589
Phone/fax: (608) 877-0909

More information on Power Glove Interfacing and sources can be found in the July 1990 issue of *Byte* magazine

I R S

401 Very Useful
402 Moderately Useful
403 Not Useful

Oscillators Don't, Amplifiers Do!

FEATURE ARTICLE

Mark Nurczyk, P.E.

A problem common to any analog circuit that uses feedback is unwanted oscillations due to an unstable design. Automated tools can help point out a circuit's weak spots and often suggest a course of action.

The title is a tongue-in-cheek saying that has been around for as long as I have been involved with electronics. Unintended oscillations are possible whenever you design high-gain analog circuits. The fear of oscillation, paired with little-known analog design techniques, keep many engineers from designing analog circuits. The simple techniques I develop here will help you get over that fear so you can begin to design stable analog circuits.

Why do amplifier circuits oscillate? Feedback. Analog circuits often use negative feedback to produce predictable circuit performance. Negative feedback works by imparting a phase shift to the feedback signal of 180°. With negative feedback, the circuit will have a predictable closed-loop performance. If the feedback network or the amplifier adds an additional 180" phase shift, the feedback will change from negative to positive. With positive feedback, the circuit will oscillate when the gain of the circuit exceeds unity. The following classic feedback equation shows why circuits oscillate:

$$A_{cl} = \frac{A_{ol}}{(1 + (A_{ol} \times B))}$$

A_{cl} = closedloop gain
A_{ol} = open loop gain
B = feedbackfactor

The ***closed-loop gain*** is the actual gain produced by the amplifier and its feedback network. The ***open-loop gain*** is the raw gain produced by the amplifier element of the circuit. For many common op-amps, the open-loop gain is approximately 100,000. The ***feedback factor*** is the reciprocal of the feedback network's transfer function.

All three elements of the feedback equation are ***phasors.*** At a given frequency, any voltage (or current) is characterized by two parameters: its magnitude and its phase shift. The mathematical representation of the magnitude and phase shift is known as a phasor, which is a dimensionless number at DC, but has magnitude and phase shift whenever the signal has an AC component. Phasor notation provides a simple method of solving tedious algebraic calculations.

If the product of the open-loop gain phasor and the feedback factor phasor equal -1, the denominator of the feedback equation shown above becomes 0. Any number divided by 0 is undefined, however we know from calculus that the limit of any number divided by 0 is infinity.

When the gain of a circuit reaches infinity, it will oscillate. In phasor notation, a quantity with a value of -1 has an absolute value of +1 and a phase shift of -180". The phase shift responsible for oscillation can come from A_{ol}, B, or both.

The criteria for stability have become rules of thumb. For absolute stability, the phase shift of the feedback signal should not exceed ± 120" (defined as a phase margin of 60") whenever the gain of the feedback signal exceeds unity. Some circuits will never have this much stability. Many designs will be stable if the phase shift does not exceed ±135° (defined as a phase margin of 45°). If the feedback phase shift exceeds ±180°, circuits with gains less than one will

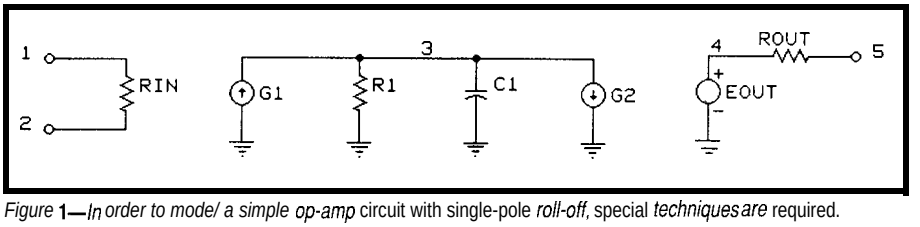


Figure 1—In order to mode/ a simple op-amp circuit with single-pole roll-off, special techniques are required.

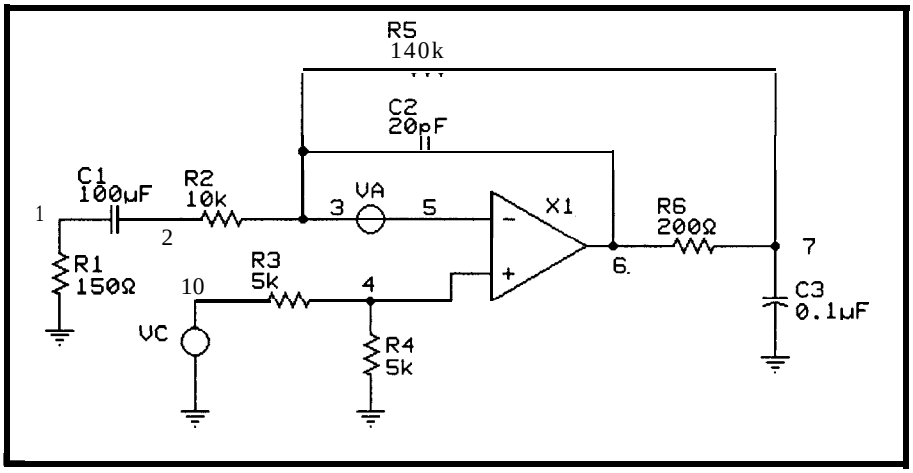


Figure 2—*Depending* on component *selection, a simple amplifier circuit can be made to behave differently. A modeling* program *such as* PSpice *makes* experimenting with *values easy.*

still be stable. For typical applications, when the phase shift of the feedback exceeds ±180°, the circuit gain should be **-12** dB or less.

During the design stage of a project, you usually want to determine a circuit's stability. A theoretically stable circuit may oscillate when breadboarded, which typically means there is a layout error. Some op-amps will oscillate with capacitive loads, but will still show theoretical stability. Understanding the theoretical performance of a circuit may save you days at the workbench.

There are many ways to determine circuit stability. Derive a couple of thousand phasor diagrams, each at a different frequency, to determine gain and phase relationships at each frequency. While this is a thorough approach, it is tedious, and it's posssible you may miss the frequency range where a problem exists.

Bode plots can be used to judge a circuit's stability. Plot both the open-loop gain of the amplifier and the feedback network's response on the same Bode plot. The slope change from one plot to the other, at the point of intersection, must be less than 12 dB per octave for absolute stability.

A pole-zero response can also be performed. If all the poles of the frequency response lie in the left half of the complex plane, the circuit is stable.

Correct circuit evaluation is possible with all of the above methods. They are tedious and require the circuit designer to have a great deal of skill. The advent of the personal computer has produced easier, faster methods. The easiest way to determine circuit stability is to use a circuit analysis program such as PSpice by MicroSim. The student edition of PSpice contains an AC analysis that determines both magnitude and phase at any frequency. An AC voltage source placed in your circuit's feedback path and swept over a large range of frequencies can show where the circuit is potentially unstable.

The circuit you are most likely to check for stability will probably involve an op-amp, so an accurate op-amp model must exist before a stability analysis can be performed. The student edition of PSpice has some restrictions on circuit size; the models for elements such as op-amps must be relatively modest, but they can still contain enough information to be useful.

Figure 1 shows a simple op-amp model with a single-pole roll-off. Generally speaking, complex parts such as op-amps require special modeling techniques. To simulate correct circuit performance, input impedance, frequency response, slew rate, voltage gain, and output parameters all have to be specified.

R_{IN} is the op-amp's input impedance as defined on the data sheet for the device and is connected to the input nodes (1 and 2). For bipolar op-amps operating at high ambient temperatures, current sources should be added from each input node to ground. These current sources simulate the input bias currents of the op-amp. The bias currents can cause

Listing 1--The amplifier *in Figure 2 can be analyzed by* writing *a model for* PSpice.

```
AC  stability  analysis
R1        1 0     1 5 0
C1        1   2     100U
R2        2    3    10K
R3        10   4    5K
R4        4    0    5K
R5        3    7    140K
R6        6    7    200
C2        3    6    20P
c3        7     0   .1U
vc   10   0    DC   5
VA   3    5    AC   1
X1        4      5      6      LMC660
.AC DEC  20    1    10E6
.PROBE
* OPAMP MACROMODEL SUBCIRCUIT
.SUBCKT LMC660          12         5
*                                  | |  |
*                                  | |  +-OUTPUT
*                                  | +-INVERTING   INPUT
*                                  +-NONINVERTING   INPUT
RIN 1    2   1T ;          INPUT          IMPEDANCE
* GAIN AND PHASE CONTROL
G1   0    3     TABLE {V(1,2)}(-0.125,-0.125 0.125,0.125) ;  SLEW RATE 1.1V/
us
R1             3   0   100000  ;  GAIN  100K
CP        3  0  1136811 ;  UNITY  GAIN  FREQUENCY  1.4MHz
G2   3    0   TABLE  {V(3)}(-0.000001,-0.125  0.0  5.0.0 5.000001.0.125)
* G2 GIVES 0.1 pS DELAY
* OUTPUT SECTION
EOUT4     0     TABLE  {V(3)}(0,0  5.5)     5 VOLT POWER SUPPLY
ROUT4     5     50.9 ; OUTPUT  RESISTANC   = 50.9 OHMS
.ENDS
.END
```

appreciable errors, especially if the input and feedback resistors have high values.

G1 is a voltage-controlled current source with a gain of 1, controlled by the voltage across R_{IN}. *G1*, in conjunction with *R1* and Cl, sets the voltage gain and frequency response of the op-amp. The value of *R1* is set to be numerically equal to the open-loop gain of the op-amp. The value of *C1* is determined by the unity gain cutoff frequency of the op-amp and is found by solving the following equation:

$$C1 = \frac{1}{(2\pi \text{ (Unity Gain Cut off Frequency)})}$$

The maximum and minimum values of *G1* can be limited to model the op-amp's slew rate. The classic capacitor equation is:

$$i = C \times \frac{dV}{dT}$$

$\frac{dV}{dT}$ = slew rate of op amp

$C = C1$

The current *(i)* is the limiting value of *G1* needed to properly model the op-amp's slew rate.

E_{OUT} is a unity gain voltage controlled voltage source controlled by the voltage across *R1*. E_{OUT} can be limited to model the op-amp's output voltage saturation characteristics. E_{OUT}, in combination with R_{OUT}, sets the output drive and resistance characteristics of the op-amp. R_{OUT} is found by using the op-amp's output voltage swing specification and is in series with the load resistance, so forms a voltage divider with the load. The value of R_{OUT} is determined by solving the following formula:

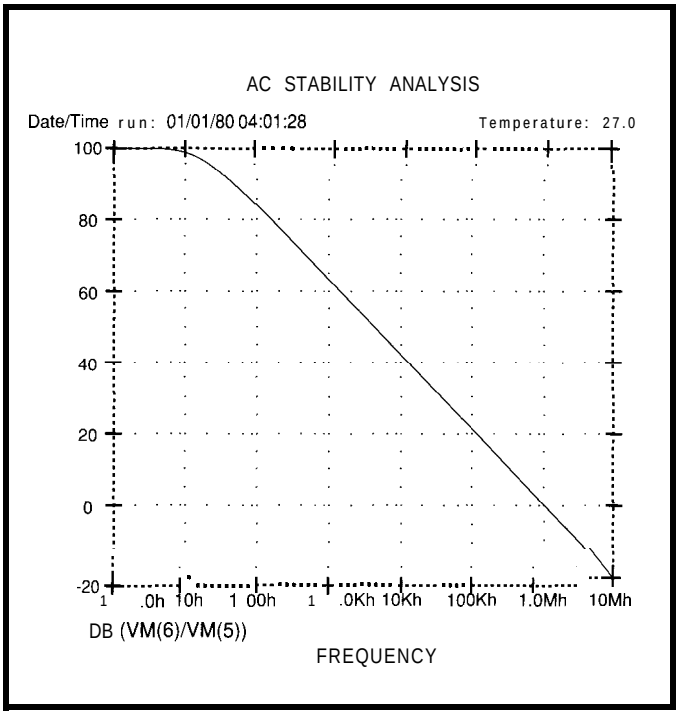


Figure 3—*The* classic single-pole *frequency response of the LMC660 op-amp as determined by PSpice matches the part's data sheet very closely.*

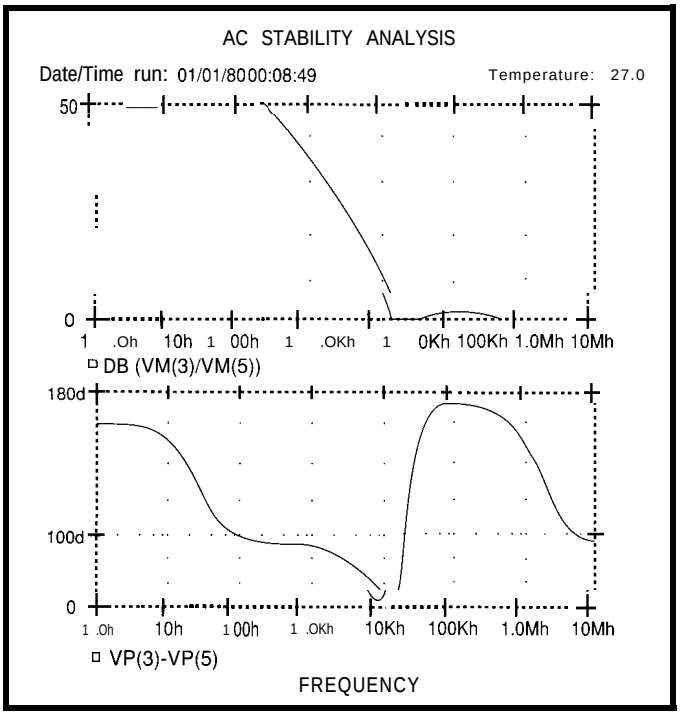


Figure *4-At* 63 *kHz,* the *phase response climbs to 173°, leaving a phase margin of only 7" and indicates a potentially unstable circuit.*

$$R_{OUT}=\frac{((V_{SUPPLY}\times R_{LOAD})-(V_{OUT}\times R_{LOAD}))}{V_{OUT}}$$

G2 prevents the voltage on node 3 from raising too high. When the voltage limit is reached, G2 generates a current with the opposite magnitude of *G1G2'*s current prevents any further voltage drop across *R 1.* Selecting the turn on voltage of G2 to be greater than the limiting voltage of E_{OUT} will model the propagation delay of the op-amp. Choosing node 3's limiting voltage to be **1** volt larger than E_{OUT}'s will produce a delay of 1 µs if the slew rate of the op-amp is **1** V/µs.

Listing **1** is the PSpice input file for Figure 2. The subcircuit for the LMC660 was made using the techniques defined above. The AC voltage source (VA) is inserted into the circuit to perform the stability analysis. The analysis is performed by sweeping VA from 1 MHz to 10 MHz. The amplitude of VA is kept small to simulate a noise source and not affect the circuit much. There are four equations that we will use to analyze the performance of Figure 2. They are:

Op-amp open-loop gain:

$$DB\left(\frac{VM(6)}{VM(5)}\right)$$

Op-amp phase response:

$$VP(6)-VP(5)$$

Feedback loop gain:

$$DB\left(\frac{VM(3)}{VM(5)}\right)$$

Feedback loop phase:

$$VP(3)-VP(5)$$

To see how good the op-amp model is, I used the first equation to produce Figure 3, which shows the classic single-pole frequency response. Comparing the curve of Figure 3 to the same curve on the LMC660 data sheet shows a very close approximation of the frequency response plot of an LMC660 op-amp.

To determine circuit stability, I made Figure 4 using the last two equations. This circuit is potentially unstable. At 63 kHz, the phase response climbs to 173". This is a phase margin of only 7" and violates the rules of thumb stated above. A lot of the excess phase shift comes from C3, which models the capacitance found in many twisted-wire-pair cables. Some method of neutralizing C3 must be found.

I made Figure 5 with C2 set to 1500 pF. The phase peak shifted to 2.8 kHz and the phase response was 147". This phase margin of 33" may keep the circuit stable, but it is still shy of the 45" defined as the minimum required. Figure 5 is the best performance that

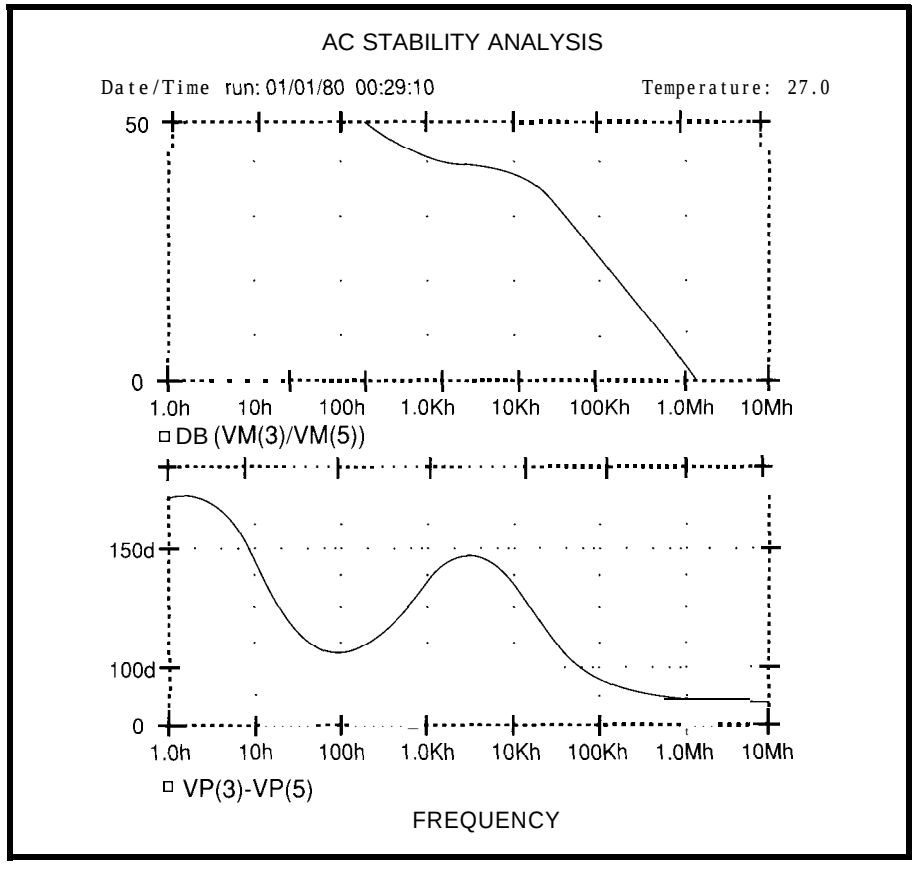


Figure 5—*Changing C2 to 1500 pF results in a marginally stable circuit, but is still not good enough*

can be realized with this circuit topology.

For Figure 6, I modified the circuit topology by reconnecting C 2 from Node 3 to ground and raising its value to 1 µF. The results show that the circuit is now unconditionally stable.

TRAILING EDGE

While the circuit shown in Figure 2 may not be the most useful op-amp circuit ever created, it has been useful to explain some very powerful design techniques. These techniques can be used with any arbitrary circuit stabilized by negative feedback. Just place the AC voltage source between the summing junction of the feedback and input network and the gain stage. ❑

Mark Nurczyk is a Registered Professional Engineer with 21 years experience in analog and digital design.

IRS

404 Very Useful
405 Moderately Useful
406 Not Useful

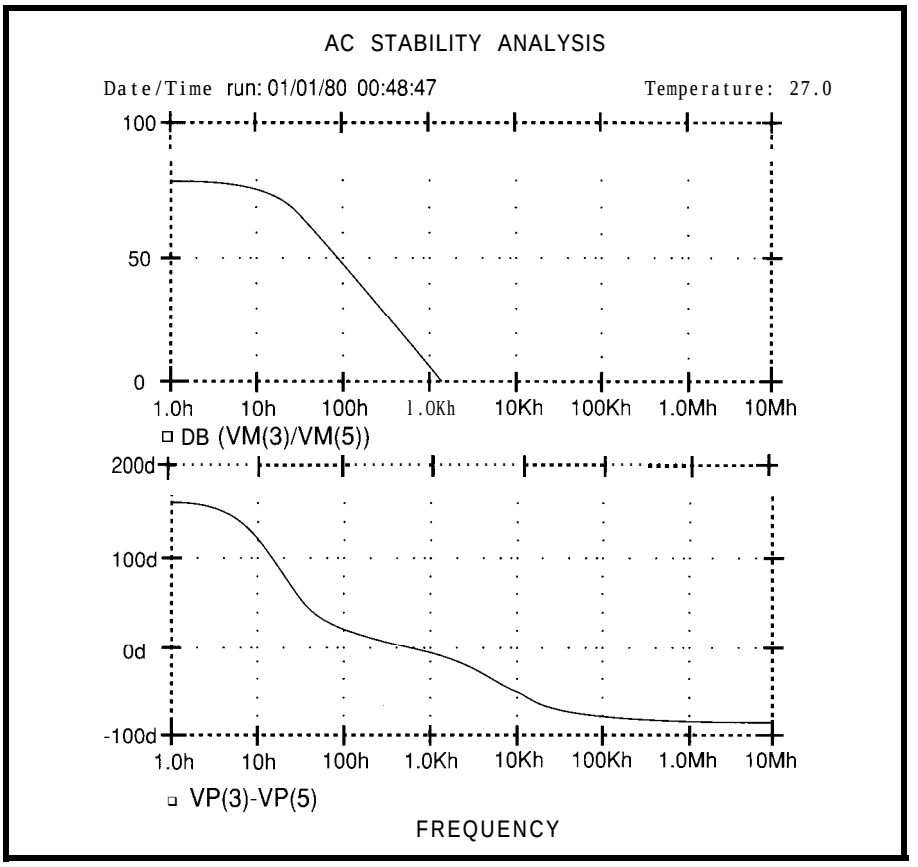


Figure 6—*With C2 connected from node 3 to ground, the circuit is unconditionally stable.*

Real-world Macintosh

A Mac SCSI interfacing primer

For all those who think the Macintosh isn't a viable platform for interfacing with the real world, they haven't explored the use of its SCSI bus. Join Marc as he begins to explore the world of Mac-based SCSI.

FEATURE ARTICLE

Marc Bumble

There are plenty of hardware design projects centered around the parallel ports of the IBM PC and IBM compatibles. Therefore, PC compatibles have been the machines of choice for hardware projects. However, the system software and the user interface available on the Apple Macintosh computers make them an attractive alternative platform for computer automation applications. In this article, I will present a first step towards uniting the Macintosh with user-created peripheral projects.

THE MACINTOSH INTERFACE

This article presents a parallel interface connected to a Macintosh SE. The parallel interface resides on a breadboard connected to the SE via the Small Computer System Interface (SCSI] port. The SCSI protocols virtually demand that the target device on the SCSI bus contain a microcontroller or some embedded logic to participate in the control of the SCSI bus. The system I present here will support embedded controllers attached to the Mac since it is designed to allow the Macintosh to download code to a microcontroller or a PROM during testing and development of your peripheral.

The SCSI blind interface I describe here can be used as a gateway to a Macintosh host. You can attach functional modules to this port to produce the following peripherals:

- EEPROM programmers
- *Microcontroller development systems
- *General data collection devices
- •Control systems

Here is a suggested order of attack to implement an embedded controller attached to the Mac via the SCSI bus:

- •Create the prototype of the intelligent target
- *Write a downloader/programmer for a microcontroller or PROM
- *Write a SCSI bus control program for the interface

For the balance of this article, I'll assume that the breadboard is the only target device on the SCSI bus, and that all data to be downloaded to the target resides in the Macintosh's RAM or on a floppy diskette. The machine's hard drive cannot be accessed because it, too, is connected to the SCSI bus. And since the target is not intelligent enough yet to obey the SCSI protocols, it will likely violate the protocols, thus rendering the hard drive inaccessible.

In this article, I will present a rudimentary SCSI test circuit and the software used to drive this hardware. The project was built and tested using a Macintosh SE. I cannot guarantee it will work with other models, however I took care to make the code portable to other Macintosh models.

EXPERIMENTAL SETUP

The easiest method of learning about SCSI is to examine the 5380 SCSI interface chip. This chip is manufactured by several vendors including NCR and National Semiconductor. To aid in user feedback, I used IO-segment bargraph displays mounted in 20-pin DIP sockets. I also added DIP switches to control the 5380's port, control, and address lines. Figures **1** and 2 show the schematic of my test bed. It allows me to control address and data lines so that I can fully test the interface chip's features and functions.

On this first go-around, the circuit is set up so you must manually control each of the 5380's processor bus lines, which means flipping switches on and off in a very specific order (that 1'11 describe as I go along). Once you're comfortable with how the chip works, you can add more intelligence (such as

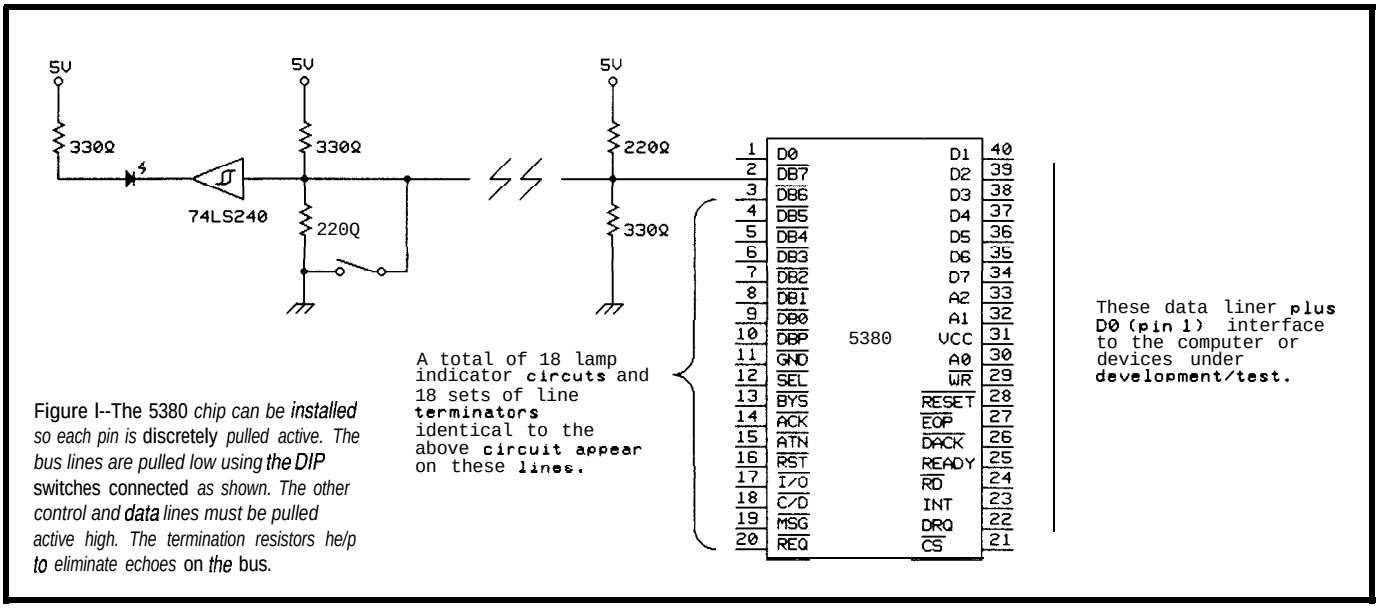


Figure I--The 5380 *chip can be installed so each pin is* discretely *pulled active. The bus lines are pulled low using the DIP* switches connected *as shown. The other control and data lines must be pulled active high. The termination resistors he/p to eliminate echoes* on *the* bus.

a processor] to automatically control the bus lines. Take special note that I used the 40-pin DIP version of National Semiconductor's DP5380N. Other packages may use different pinouts..

THE 5380 REGISTERS AND CONTROLS

Three registers must be set in order to read from and write to the 5380 (see Figure 4): the Output Data Register (ODR), the Initiator Command Register (ICR), and Mode Register 2 (MR2). The registers are accessed by using address lines AO, Al, and A2, which are active high. To access MR2, for example, set A0 and A3 low and pull Al high. To access the ODR, pull all three lines low. The eight bits in each register are individually set using data lines DO-D7.

The 5380 is described in the Mass Storage Handbook published by National Semiconductor. Those of you interested in doing further development with the chip can find a complete description of the device in that book.

SETTING THE 5380 REGISTERS

First, I will present the general method of setting the 5380 registers, then I'll give a specific example of how to set the registers to allow data to be written out to the SCSI bus.

To set a register, first set ● CS high (inactive). Next, set the address lines to the binary address corresponding to the desired SCSI register. Then set the data lines with the information to go to the register. Enable the chip by bringing *CS low. Finally, ensure *RD is high and pulse *WR low to transfer the data into the register.

In terms relevant to the SCSI bus standard, the *initiator* is a device that assumes control of the bus. There can be only one initiator at any given time. The *target* is any other peripheral connected to the bus. The SCSI standard allows up to seven target devices and one initiator on each SCSI bus.

For my project, I'll assume that there is only one target connected to the bus (the breadboard), with the initiator being the Macintosh. This assumption allows me to use the SCSI interface without having to select which of the seven possible target peripherals is desired. By assuming that the breadboard is the only listener on the bus, I can have more control over how I manipulate the data and control lines. My entire test bed is illustrated in Figure 3.

In the final version of this project, all the register manipulations necessary will be carried out via the com-

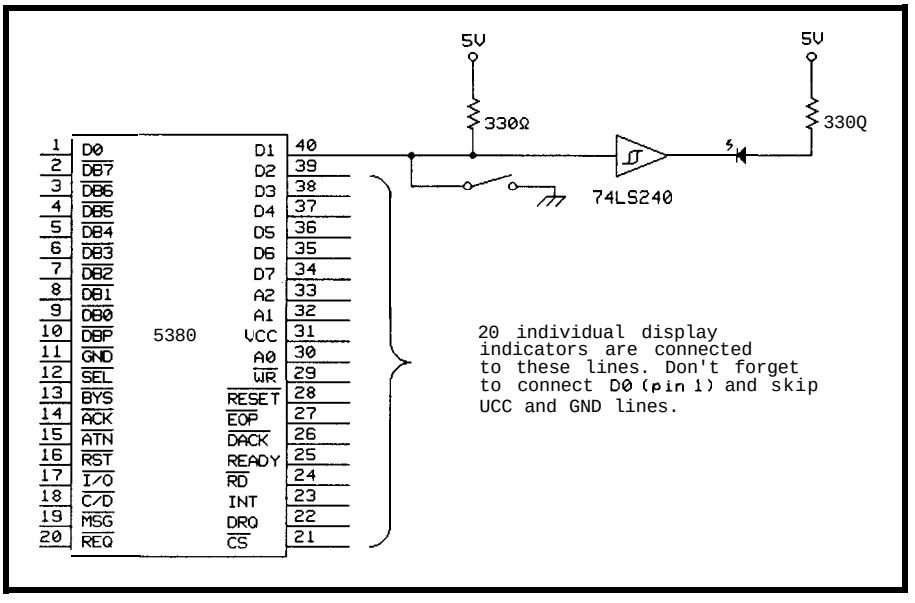


Figure *2-The SCSI control/data indicator circuit is similar to the previous setup for the SCSI* bus lines, however *a* single pull-up *resistor has been substituted for the bus termination dual resistor setup. The DIP switch is used to maintain the control data lines at ground potential.*

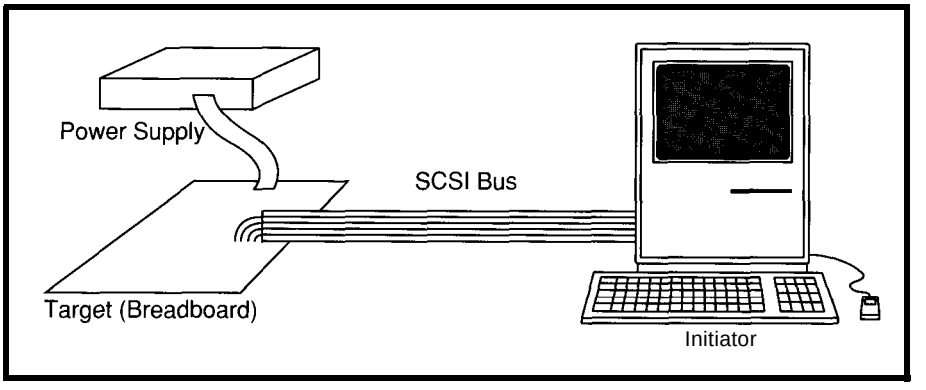


Figure 3-The target *breadboard is the Mac's link to real-world signals. In the final implementation, the target will need some intelligence (a processor or PAL) to control the* 5380 *and automatically direct raw data onto the SCSI* bus.

Output Data Register (ODR)

Bit 7							Bit 0
DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0

8 Bits Hex Addr 0 Write-Only

Initiator Command Register (ICR)

Bit 7							Bit 0
RST	TEST	LA/DIFF	ACK	BSY	SEL	ATN	DBUS

8 Bits Hex Addr 1

Mode Register 2 (MR2)

Bit 7							Bit 0
BLK	TARG	PCHK	PINT	EOP	BSY	DMA	ARB

8 Bits Hex Addr 2 Read-Write

Current SCSI Data (CSD)

Bit 7							Bit 0
DB7	DB6	DB5	DB4	DB3	DB2	DB1	DBO

8 Bits Hex Addr 0 Read-Only

Target Command Register (TCR)

Bit 7							Bit 0
X	X	X	X	REQ	MSG	CID	I/O

8 Bits Hex Addr 3 Read-Write

Select Enable Register (SER)

Bit 7							Bit 0
DB7	Df36	DB5	DB4	DB3	DB2	DB1	DBO

8 Bits Hex Addr 4 Write-Only

Current SCSI Bus Status (CSB)

Bit 7							Bit 0
RST	BSY	REQ	MSG	C/D	I/O	SEL	DBP

8 Bits Hex Addr 4 Read-Only

Bus and Status Register (BSR)

Bit 7							Bit 0
EDMA	DRQ	SPER	INT	PHSM	BSY	ATN	ACK

8 Bits Hex Addr 5 Read-Only

Figure *4-The* 5380 *SCSI interface chip has eight registers that are used to communicate with the host processor. Only three of them are necessary for very basic experimentation.*

puter by the machine's internal SCSI controller chip. To become familiar with the operations of this chip, it is best to experiment with it while it is in this no-holds-barred breadboard setup. From the peripheral side of the 5380, the correct pins must be set to place data on the SCSI bus.

FOR EXAMPLE

Now, I'll present a specific example of how to set the registers to allow data to be written out to the SCSI bus. Like any project combining software and hardware, the board must be initialized to a known state before anything predictable and useful can happen, which means all of the control, signal, and data lines should be set to their floating, or off, states. The DIP switches should be set so that all of the bus lines are floating at 3.33 volts (all should be open). The data and control lines need to be set to their inactive state, which means the DIP switches for those need to be closed. The switches that control the address lines (AO, Al, and A2) should also be closed. The DIP switches covering

- WR, 'RESET, *EOP, *DACK, 'RD, and *CS should be left open. Finally, READY, INT, and DRQ should be switched to ground.

Once the 5380 is set in its initial state, the next step is to configure the chip to place data on the SCSI bus using MR2, ICR, and ODR [see Figure 4].

The first step in setting the 5380 registers is to set ICR bit 3 (BSY). ICR is located at offset 1 and is shown in detail in Figure 5a.

After the ICR BSY line is set, bit 6 of MR2 (offset 2) is enabled. All other bits in MR2 are disabled. The address, control, and data pins must be set as shown in Figure 5b.

Once the pins are set up, click the * CS pin momentarily over to the "0" state to enable the data in MR2. After MR2 is set, the ICR settings can be configured. The DBUS bit must be set to enable the contents of ODR onto the SCSI bus data lines. The parity bit, DBP, will also be automatically generated by this operation. To set this register, configure the DIP switches as shown in Figure 5c.

Once again, momentarily switch *CS to "0" in order to write the data to the register. To set data onto the bus, simply write data to the ODR. The pin settings for this operation are shown in Figure 5d.

When 'CS is set to "0," the data indicated by the DBO-DB7 lines will be flushed out onto the bus. You can leave ● CS set to "0" and use the DB0--DB7 lines to change the data on the bus.

Of course, this presentation is not the standard manipulation of the SCSI control lines and protocol, but instead it serves to illustrate the basic operations of a SCSI communications device. For a full implementation of a SCSI device, we need some intelligence provided by a processor or a PAL to control the 5380.

CONNECTING THE MACINTOSH TO THE BREADBOARD

Make sure the cable is carefully constructed, since improperly constructed SCSI cables have been known to permanently disable a Mac motherboard. In the creation of my project, I soldered short extensions onto the cable (about 1.5 inches) to allow the individual lines of ribbon cable to be easily inserted into the breadboard. The cable construction is detailed in Figure 6. The plug used to connect to the SCSI port on the Macintosh is a "male D-dubminiature 25-pin" connector. The plug signal assignments are detailed in Figure 7.

The sample driver code provided is written in 68000 assembly language and is used to place bits into the SCSI data registers. Two separate code segments are provided: one for reading and the other for writing to the data bus.

The code presented is designed purely for testing the interface and the breadboard, so the first step is to reset and initialize the SCSI bus and then pause for the user to reset and initialize the test breadboard. The call to `GetNextEvent` simply waits for a mouse or keyboard event which allows time for the user to initialize the breadboard. Next, the MacsBug debugger is summoned to let the user single step through the code. The

a)	A0	=	1	Sets the register to hex address 1	**b)**	A0	=	0	Sets the register to hex address 2
	A1	=	0	Initiator Command Register		A1	=	1	AO-A2 => 010 = 2; See Figure 4
	A2	=	0	See Figure 4		A2	=	0	Mode Register 2
	DBO	=	0	Deassert ODR data on bus data lines		DBO	=	0	Disable Arbitration
	DB1	=	0	Deassert ATN		DB1	=	0	Disable DMA mode
	DB2	=	0	Deassert SEL		DB2	=	0	Disable BSY Monitor
	DB3	=	1	Enable BSY BSY will remain on		DB3	=	0	No interrupt for End of Process (EOP)
	DB4	=	0	Deassert ACK		DB4	=	0	No interrupt for parity error
	DB5	=	0	Arbitration is NOT enabled		DB5	=	0	No SCSI parity checking
	DB6	=	0	Normal Mode		DB6	=	1	Set to Initiator Mode
	DB7	=	0	Deassert RST		DB7	=	0	Non-Block DMA
	WR	=	0	Allows the register to be set.		WR	=	0	Allows the register to be set
	RD	=	1			RD	=	1	
c)	A0	=	1	Sets the register to hex address 1	**d)**	A0	=	0	Sets the register to hex address 0
	A1	=	0	Initiator Command Register		A1	=	0	The Output Data Register
	A2	=	0	See Figure 4		A2	=	0	
	DBO	=	1	Enable ODR data on bus data lines		DBO	=	1	Set these data lines to the desired state to
	DB1	=	0	Deassert ATN		DB1	=	0	place the data byte on to the SCSI bus.
	DB2	=	0	Deassert SEL		DB2	=	0	
	DB3	=	0	Enable BSY BSY will remain on		DB3	=	0	The data sent to the bus can be controlled
	DB4	=	0	Deassert ACK		DB4	=	0	by the CS line
	DB5	=	0	Arbitration is NOT enabled		DB5	=	0	
	DB6	=	0	Normal Mode		DB6	=	0	
	DB7	=	0	Deassert RST		DB7	=	0	
	WR	=	0	Allows the register to be set		WR	=	0	Allows the register to be set
	RD	=	1			RD	=	1	

Figure *5-A basic write to the SCSI bus through the* 5380 *includes (a) setting the Initiator Command Register (ICR) in prepartion for setting MR2, (b) setting the Mode Regsiter 2 (MR2), (c) setting the ICR again to enable data onfo the bus, and (d) setting the actual data in the Output Data Register (ODR).*

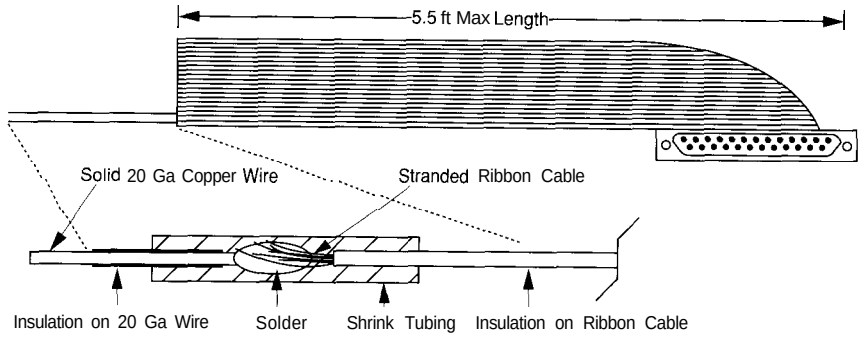


Figure 6—*Proper construction of the connection cable will help lead to successful experimental results. Twenty-gauge wire allows for easy insertion into the breadboard.*

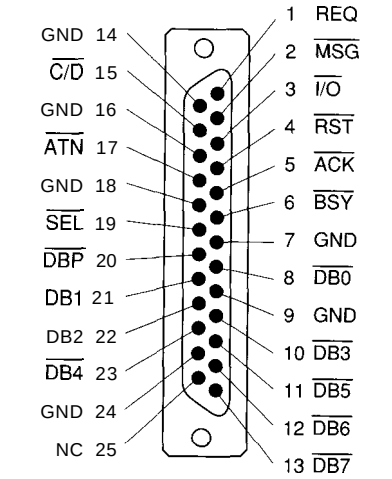


Figure *7-The Mac puts the typically 50-pin SCSI interface on a standard 25-pin D-type connector.*

Listing I-Using *a mix of C and assembler, the write test code simply sends out an alternating bit pattern on the SCSI bus.*

```
i/include <stdio.h>              /*        SCSIWR Test Code        */
i/include <sane.h>
    OSErr result:
#define SCSIBase      0x0C00  /* Output Data Register with RACK */
#define SCSIGlobals   0x0C0C  /* Current SCSI Data with DACK    */
#define dackWr        0x0201  /* Output Data Register with DACK */
#define dackRd        0x0260  /* Current SCSI Data with DACK    */
#define sODR          0x0001  /* Output Data Register           */
#define sWrICR        0x0011  /* Initiator Command Register     */
#define sWrMR2        0x0021  /* Mode Register 2                */
#define sWrTCR        0x0031  /* Target Command Register        */
#define sSER          0x0041  /* Select Enable Register         */
#define sDMAtx        0x0051  /* Start DMA Send                 */
#define sTDMArx       0x0061  /* Start DMA Target Receive       */
#define sIDMArx       0x0071  /* Start DMA Intiator Receive     */
#define sCDR          0x0000  /* Current SCSI Data              */
#define sRdICR        0x0010  /* Initiator Command Register     */
#define sRdMR2        0x0020  /* Mode Register 2                */
i/define sRdTCR       0x0030  /* Target Command Register        */
#define sCSR          0x0040  /* Current SCSI Bus Status        */
#define sBSR          0x0050  /* Bus and Status Resgister       */
#define sIDR          0x0060  /* Input Data Register            */
#define sRESET        0x0070  /* Reset  Parity/Interrupt        */

main0

  EventRecord event:
  printf("SCSI test in Progress. Click mouse to continue.\n\n");
  printf("This program requires 'Macsbug' be installed.\n\n");
  while (!GetNextEvent(mDownMask + keyDownMask, &event)) {};
  result = SCSIReset();
  TestResult();
  printf("Click mouse to continue.\nUse 's' to step.\n ");
  while (!GetNextEvent(mDownMask + keyDownMask, &event)) {};
  asm {
  Debugger:  /*    Invoke  Macsbug  debugger  */
  MOVEA.L  (A7)+,A0
  LINK     A6,#0xFFF8
  MOVEM.L  D2-D7/A2-A4,-(A7)
  MOVEA.L  SCSIBase,A3              ; SCSIBase
  MOVEA.L  SCSIGlobals,A4           ; SCSIGlobals
  LEA      sWrICR(A3),A0            ; Set A0 to point to the
                                    ; Initiator Command Register
  ;1) Set the BSY line to active low. This line is set by
  ; accessing the Initiator Command Register (HA 1 of the 5380
  ; chip). The required bit setting is the 4th bit or the BSY bit.
```

(continued)

system operation can be verified by observing the LEDs on the breadboard.

The SCSI base address is stored in register A3, the SCSI Global Parameters address is moved into A4, then we begin the 5380 bit manipulations. We start by setting A0 to point to ICR using an LEA (Load Effective Address] call. Once the ICR address is established, the BSY line can be set active low through the fourth bit (DB3) in the ICR.

Next, the MR2 target bit is pulled low to set the chip in its target mode. In target mode, only the target mode bit and the ICR DBUS bit need to be set to place data onto the bus. So the next function performed is to load the ICR address and set the DBUS bit, which is bit 1 (or DBO). Once the DBUS bit is set to active low, data can be moved out onto the SCSI bus.

Register A0 is set to point to ODR. Then, the code simply moves data into the ODR. The MO V E . B command moves bytes of data out onto the bus. I selected the patterns AA and 5 5 because they are viewed in binary as:

10101010	AA
01010101	55

The patterns of alternating LEDs should be evident as you step through the code. The patterns will blur and be undetectable if the code is run at full speed. See the MacsBug Manual for instructions on how to single step through the code.

Listing *1-continued*

```
 ; The Initiator Command Register is also referred to as the ICR.

 MOVE.B  #0x08,(A0)               ; This move pulls the BSY
                                  ; line active low.
 ;2); Once the BSY bit has been set the Target Bit in the Mode
 ; Register must be pulled low.  This can be accomplished by
 ; the following code:
 LEA     sWrMR2(A3),A0            ; Set A0 to point to the
                                  ; Mode Register 2.
 MOVE.6  #0x40,(A0)               ; Pull TARG bit low

; 3) Now the ICR which was used in step 1 above is again
; called and the DBUS bit (bit zero) must be set to allow
;us to put data out on the SCSI bus. So we must reset our
; Address in register A0.
  LEA     sWrICR(A3),A0            ;Set A0 to point to the ICR
  MOVE.B  #0x01,(A0)               ; Pull DBUS bit low

; 4) We should now be able to write data out to the SCSI bus at
; will by directing our hex data stream to the Output Data
; Register.
  LEA     sODR(A3),A0              ;Set A0 to point to the ICR
  MOVE.B  #0xAA,(A0)               ; Pull ODR bits low
  MOVE.B  #0xAA,(A0)
  MOVE.B  #0x55,(A0)
  MOVE.B  #0xAA,(A0)
  MOVE.B  #0x55,(A0)
  MOVE.B  #0xAA,(A0)
  MOVE.B  #0x55,(A0)
  MOVE.B  #0xAA,(A0)
  MOVE.B  #0x55,(A0)
  MOVE.B  #0xAA,(A0)
  MOVE.B  #0x55,(A0)
  MOVE.B #0xAA, (A0)
  MOVE.B  #0x55,(A0)
  MOVE.B  #0xAA,(A0)
  MOVE.B  #0x55,(A0)
  MOVE.B  #0xAA,(A0)
  MOVE.B  #0x55,(A0)
  MOVE.B  #0xAA,(A0)
  MOVE.B  #0x55,(A0)
  MOVE.B  #0xAA,(A0)
  MOVE.5  #0x55,(A0)
  MOVE.B  #0xAA,(A0)
  MOVE.B  #0x55,(A0)
  MOVE.B  #0xAA,(A0)
  MOVE.B  #0x55,(A0)
  MOVE.B  #0xAA,(A0)
  MOVE.B  #0x55,(A0)
  MOVE.B  #0xAA,(A0)
  MOVE.B  #0x55,(A0)
  MOVE.B  #0xAA,(A0)
 UNLK    A6
 }
  result = SCSIReset();
  TestResult();
}
TestResult()
{
  switch(result)
  {
     case noErr:
        printf("SCSI noErr result.\n\n");
     break:
     default:
     break;
  }
}
```

Listing 2—*The code to test the reading function simply does continuous reads of the bus, allowing you to inspect each reading by single stepping through the program with a debugger.*

```
#include <stdio.h>              /*        SCSIRD Test Code        */
#include <sane.h>
    OSErr result;
#define SCSIBase      oxocoo    /*Output Data Register with DACK */
#define SCSIGlobals  0x0C0C     /* Current SCSI Data with DACK */
#define dackWr        0x0201    /* Output Data Register withDACK*/
#define dackRd        0x0260    /* Current SCSI Data with DACK */
#define sODR          0x0001    /* Output Data Register          */
#define sWrICR        0x0011    /* Initiator Command Register    */
#define sWrMR2        0x0021    /* Mode Register 2               */
#define sWrTCR        0x0031    /* Target Command Register       */
#define sSER          0x0041    /* Select Enable Register        */
#define sDMAtx        0x0051    /* Start DMA Send                */
#define sTDMArx       0x0061    /* Start DMA Target Receive      */
#define sIDMArx       0x0071    /* Start DMA Intiator Receive    */
#define sCDR          0x0000    /* Current SCSI Data             */
#define sRdICR        0x0010    /* Initiator Command Register    */
#define sRdMR2        0x0020    /* Mode Register 2               */
#define sRdTCR        0x0030    /* Target Command Register       */
#define sCSR          0x0040    /* Current SCSI Bus Status       */
i/define sBSR         0x0050    /* Bus and Status Resgister      */
#define sIDR          0x0060    /* Input Data Register           */
#define sRESET        0x0070    /* Reset Parity/Interrupt        */

main()
{
    EventRecord event:

    printf("SCSI test in Progress Click mouse to continue.\n\n");
    printf("This program requires 'Macsbug' be installed.\n\n");
    while (!GetNextEvent(mDownMask + keyDownMask, &event)) {};

    result = SCSIReset();
    TestResult();

    printf("Click mouse to cont.\nUse's' to step debugger.\n");
    while (!GetNextEvent(mDownMask + keyDownMask, &event)) {};

    asm {
    Debugger;                   /*Invoke Macsbug debugger       */

    MOVEA.L  (A7)+,A0
    LINK     A6,#0xFFF8
    MOVEM.L  D2-D7/A2-A4,-(A7)
    MOVEA.L  SCSIBase,A3        ; SCSIBase
    MOVEA.L  SCSIGlobals,A4     ; SCSIGlobals

    LEA      sCDR(A3),A0

; Set A0 to point to the Current SCSI Data

;1) Once the address register, A0, contains the address of the
;Current SCSI Data Register, the data can be read straight
;off the SCSI data lines.  It's up to the peripheral to place
;the data on the lines.  Other sections of the SCSI protocol
;can be implemented to determine when the data is available
;for reading.

;2) Run the compiled application through the first mouse
;click. Then, use the method presented in the article to set the
;breadboard.  The object is to imitate a peripheral placing
;data on the SCSI bus.  The user can also choose to set the
;data lines directly on the SCSI bus, avoiding the 5380 chip,
;if desired.
```

(continued)

The code in Listing 2 contains a routine for reading that is much simpler than the write routine.

CONCLUSIONS

The code presented in this article demonstrates how to access the 5380 SCSI driver chip. This access should allow further work to establish other peripheral projects for the Macintosh family of computers. Many of the design projects currently available for the IBM PC and IBM compatibles can now be established or ported to the Macintosh.

In addition to the project ideas I outlined earlier, this project can be adapted to test protocols for communications experiments between two machines. This system can also be useful for other tests wherein the Macintosh SCSI interface is used to host experiments and experimental peripherals. ❑

The work in this article is dedicated to Dr. Fred Ketterer, who teaches electrodynamics, electromechanics, and digital circuits at the University of Pennsylvania.

Marc holds a BSEE from the University of Pennsylvania, is currently finishing his MSEE and is pursuing a PHD in Computer Engineering. As a communications engineer, his specialties include RF communications systems and cellular and satellite communications networks.

REFERENCES

MacArthur, Jim, "Build a Simple SCSI-to-Anything Interface," ***Circuit Cellar INK,*** April/May 1990, p 15.

Eng, John, "Part 1: An Intelligent SCSI Data Acquisition System for the Apple Macintosh," ***Circuit Cellar INK,*** June/July 1989, p 36.

Eng, John, "Part 2: An Intelligent SCSI Data Acquisition System for the Apple Macintosh," ***Circuit Cellar INK,*** Aug./Sept. 1989, p15.

Hodges, Mike, "Part 1: The SCSI Bus," ***BYTE,*** Feb. 1990, p 267.

Hodges, Mike, "Part 2: The SCSI Bus," ***BYTE,*** Mar. 1990, p 291.

Inside Macintosh Volume IV, Addison-Wesley, Reading, MA, 1986.

Mass Storage Handbook, National Semiconductor, 1989

MC68000 *8-/16-/32-Bit* ***Microprocessor User's Manual,*** Motorola, 8th edition, Prentice Hall, Englewood Cliffs, NJ.

NCR 5380 Family, SCSI Protocol Controller, Data Manual, 1989.

SOFTWARE

Software for this article is available from the Circuit Cellar BBS and on Software On Disk for this issue. Please see the end of "ConnecTime" in this issue for downloading and ordering information.

I R S

407 Very Useful
408 Moderately Useful
409 Not Useful

Listing ***2-continued***

```
    MOVE.B  (A0),D0         ;Moves data from the SCSI bus into D0
    MOVE.B  (A0),D0         ;The last word (last 2 bytes) of D0 will
    MOVE.B  (A0),D0         ;change to reflect values of D0-07 on
    MOVE.B  (A0),D0         ;the SCSI lines.  Use Macsbug to step
    MOVE.B  (A0),D0         ;through the list of MOVE.5 commands.
    MOVE.B  (A0),D0         ;Changing the Bus data lines will cause
    MOVE.B  (A0),D0         ;the contents of D0 to change.  D0 is
    MOVE.B  (A0),D0         ;visible in Macsbug.
    MOVE.B  (A0),D0
    MOVE.B  (A0),D0
    MOVE.B  (A0),D0
    MOVE.B  (A0),D0
    MOVE.B  (A0),D0

    UNLK    A6
    }
    result = SCSIReset()
    TestResult();
}
TestResult()
{
    switch(result)
    {
        case noErr:
                printf("SCSI noErr result.\n\n")
                break:
            default:
                break:
        }
```

A Parallel Expander for the PC

FEATURE ARTICLE

John F. Lenihan

Have you ever wondered why IBM didn't do it the right way and make the parallel printer port bidirectional? With a few chips and some special software, you can make the PC's printer port into something useful.

Without special hardware, all PCs have only two ways to connect to the outside world: serial and parallel ports. The advantages and drawbacks of the serial port are well known, so I won't discuss them here. However, the parallel port, unlike the serial port, tends to be used in a predictable, fixed way for interface projects.

Everyone is familiar with printers, disks, tape drives, and scanners interfaced through the parallel port. Not so familiar, but still used, are such exotic devices as motor controllers and radiation monitors. That's terrific when all you want is the special device connected to the parallel port, but no help at all when you want to connect the parallel port to some device or system, perhaps several different ones at different times.

The Parallel Expander is the answer to that problem. Five TTL chips and a few connectors provide 16 TTL outputs, **16** TTL inputs, 2 TTL strobe pulses, and an interrupt-which is a spectacular amount of I/O for such a simple, inexpensive circuit. Figure 1 shows a block diagram of the Parallel Expander interface.

THE PARALLEL PORT

Figure 2 provides the details of the various bits, their functions, their port assignments, and whether or not they are inverted. The BASE port is assigned by the operating system to either 0278H, 0378H, or 03BCH, depending on factors like the type of monitor and the number of parallel ports present.

A notable feature of the parallel port is that while five bits are available for reading, one of these is an interrupt and is reserved for that purpose. The remaining four bits have one bit inverted and separated from the others. These inconsistencies makes for some interesting software gymnastics when reading the bits.

Another interesting feature is that the control bits (BASE+3) are all inverted but one. Experiment also showed that these bits do not all change at the same time, which can create a potential glitch problem unless considered in the design.

The Parallel Expander will not work unless all the signals of Figure 3 are present; the widespread use of parallel port interfaces, however, indicates that crippled or oddball ports are very much the exception.

THE PARALLEL EXPANDER CIRCUIT

Figure 4 shows the schematic diagram of the circuit. The input is a male DB-25 connector (P01) that mates

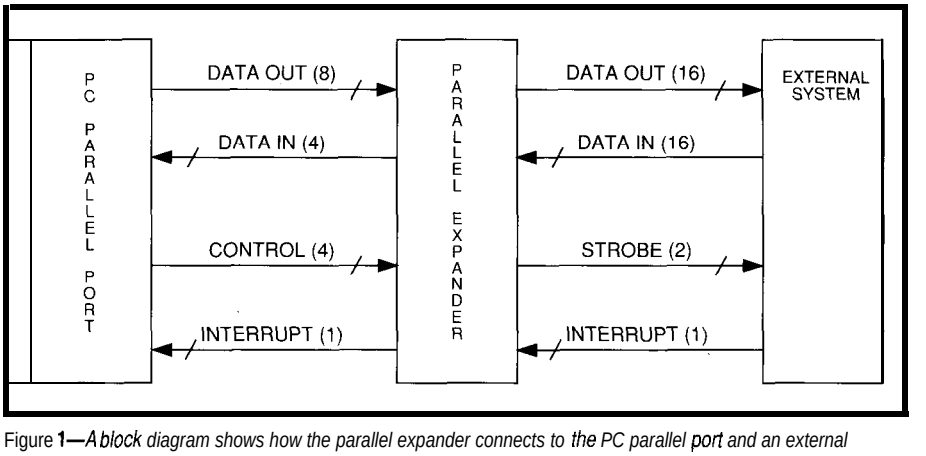


Figure 1—*A block diagram shows how the parallel expander connects to the PC parallel port and an external system.*

with the standard parallel port connector. The data bits are applied to both octal D-type flip-flops (Ul and U2) and are read in eight at a time by the decoding circuit, to give 16 output bits (OUT-O through OUT-15). The 16 input bits (U4 and U5) are selected four at a time by the decoder and sent to the parallel port.

The decoder chip (U3) is the key to stable, glitch-free operation of this circuit. Essentially, the decoder is set up with the right address to perform a particular function and then strobed by the next computer instruction to execute the function. Figure 3 will make decoder operation clear.

SOFTWARE

In order to monitor and control all the extra "tentacles" provided by the Parallel Expander, of course the right software is needed.

Listing 1 shows a few of the routines contained in PARX FAST. UN I, which is the heart of the software and is written as a Turbo Pascal 6.0 unit. The procedures to set the printer port, as well as some of the bit setting and testing routines, are in Pascal. The critical routines (shown in the listing) are in assembler, which dramatically increases the speed and reduces the size of the code.

Refer to Figures 2 and 3 when reading the assembler code; the tables will help you understand how the software deals with the gap in the input bits and the addressing and strobing process for the decoder. Note that all I/O is done 16 bits at time. It's not much trouble to change this for fast 8-bit I/O; the code is already there-just rearrange it.

The software should be very easy to recast in all-assembler, C, other versions of Pascal or BASIC. I would expect, however, that using interpreted BASIC will cause a drastic slowdown in execution.

CONSTRUCTION AND TESTING

The prototype of my project was wire-wrapped. Almost any layout will work as long as it is neat and adequately bypassed. A metal box is the ideal enclosure, even for units built on insulating surfaces such as fiberglass.

DB-25 pin	Name	port	Bit	Details
1	*Strobe	Base+2	0	Read/Write; inverted
2-9	Data0–7	Base	o-7	Write only; BASE port
10	ACK	Base+1	6	Read only; causes INT if grounded
11	*Busy	Base+1	7	Read only; inverted
12	PE	Base+1	5	Read only
13	Select	Base+1	4	Read only
14	*Autofeed	Base+2	1	Read/Write; inverted
15	Error	Base+1	3	Read only
16	Init	Base+2	2	Read/Write; BASE+2 = control bits
17	*Select	Base+2	3	Read/Write; inverted

Figure 2—*The "standard" IBM PC printer port connections are fairly well defined. All undefined* **pins** *should be connected to ground.*

CTL Port Bits B3 B2 B1 B0	Output to Decoder ● C3 C2 *C1 *C0	U3 pin	Function	CTLPort Seq for Execution
0 0 0 0	1 0 1 1	Y7 (07)	EXT_STB_1	0 -> 4 -> 0
0 0 0 1	1 0 1 0	Y6 (09)	EXT_STB_0	1 -> 5 -> 1
0 0 1 0	1 0 0 1	Y5 (10)	WRT_LOBYT	2 -> 6 -> 2
0 0 1 1	1 0 0 0	Y4 (11)	WRT_HIBYT	3 -> 7 -> 3
1 0 0 0	0 0 1 1	Y3 (12)	RD_NIBL_2	8 -> c -> 8
1 0 0 1	0 0 1 0	Y2 (13)	RD_NIBL_3	9 -> D -> 9
1 0 1 0	0 0 0 1	Y1 (14)	RD_NIBL_0	A -> E -> A
1 0 1 1	0 0 0 0	Y0 (15)	RD_NIBL_1	B -> F -> B

Figure 3—*Each output bit is accessed in a unique way. In each case, set up the decoder with B3, B1, and B0. Then raise B2 to execute the function (same as adding 4). Finally, lower B2 to end the execution.*

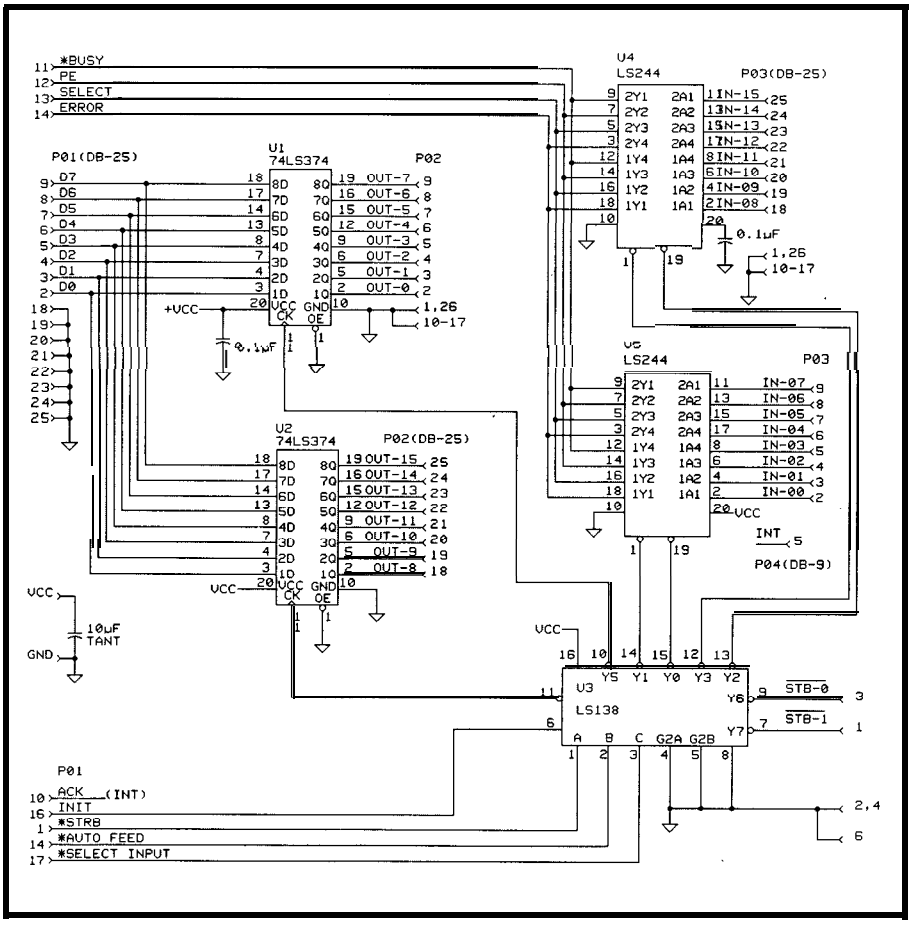


Figure 4-*The parallel expander uses two flip-flops to read in data 8 bits at a time. The decoding circuit eventually sends sixteen input bits (four at a time) to the parallel port.*

Listing 1—*Utility routines for the Parallel Expander are written as a Turbo Pascal 6.0 unit, however the more critical routines are done in assembler. Complete code is available on the BBS.*

```
PROCEDURE GETPORT(LPTNUM:WORD);
BEGIN
    IF (LPTNUM<1) OR (LPTNUM>4) THEN LPTNUM:=2;
    {LPT2 IS DEFAULT}
    OUTPORT:=MEMW[$0040:$0008+2*(LPTNUM-1)];
    {LPT1 THROUGH LPT4 PORT#'S}
    {AT 0040:0008 0040:000E }
    INPORT:=OUTPORT+1;
    CTLPORT:=OUTPORT+2;
END;

PROCEDURE READWORD(VAR INW:WORD);
VAR J:WORD;X:BYTE;          {INPUT BITS: *7_543___}
{FAST CODE WITHOUT LOOPS}{OUTPUT BITS: ____3210}
BEGIN
    ASM
    XOR AX,AX; MOV BX,AX; MOV CX,04 {Set up registers}

    MOV DX,CTLPORT          {Set up decoder for nibble 0}
    MOV AL,$0A
    OUT DX,AL
    MOV AL,$0E
    OUT DX,AL               {Enable decoder to read nibble 0}
    MOV DX,INPORT
    IN AL,DX                {Read in nibble 0}
    AND AL,$B8              {Mask unused bits}
    XOR AL,$80              {Invert bit 71
    SHR AL,1;SHR AL,1;SHR AL,1      {Shift everything lower)
    TEST AL,$10             {If B4 set, clear it and set B3}
    JZ @1
    SUB AL,$08
@1  OR BX,AX                {Save nibble in BX}
    ROR BX,CL               {Set up BX for next nibble)
    MOV DX,CTLPORT          {Disable decoder)
    MOV AL,$0A
    OUT DX,AL

    MOV DX,CTLPORT          {Do nibble 1
    MOV AL,$0B
    OUT DX,AL
    MOV AL,$0F
    OUT DX,AL
    MOV DX,INPORT
    IN AL,DX
    AND AL,$B8
    XOR AL,$80
    SHR AL,1;SHR AL,1;SHR AL, 1
    TEST AL,$10
    JZ @2
    SUB AL,$08
@2: OR BX,AX
    ROR BX,CL
    XOR AX,AX
    MOV DX,CTLPORT
    MOV AL,$0B
    OUT DX,AL

    MOV DX,CTLPORT          {Do nibble 2
    MOV AL,$08
    OUT DX,AL
    MOV AL,$0C
    OUT DX,AL
    MOV DX,INPORT
    IN AL,DX
    AND AL,$B8
    XOR AL,$80
    SHR AL,1;SHR AL,1;SHR AL,1
```

(continued)

The circuit runs on 5 V. The power supply is not critical and can be obtained from any source, even the seldom-used joystick port. Simply steal +5 V from pins 1, 8, 9, and 15 and ground from pins 4, 5, and 12 on the joystick port's 15-pin D-type connector. If power is obtained in this way, the Parallel Expander should not be too far from the computer-3 feet or so. The current requirement for the Parallel Expander is about 100 mA.

I also wrote a test program—PARXTEST. PAS-entirely in Turbo Pascal 6.0 (available on the BBS). The program uses the unit generated by PARXFAST.UNI todoalltheparallel I/O. It is simple and straightforward—there are no windows, shadow boxes, garish colors or anything like that—but the program works and is intuitively easy to use.

My code includes the ability to exercise the Parallel Expander hardware. The tests include: reading and writing random values, read/write timing, bit set/clear, and interrupt action. Before running the tests, connect a 25-wire cable from the outputs to the inputs (from PO3 to PO2). The test program gives any necessary instructions; for example, using a logic probe (or a 'scope) to test strobes and when to ground the interrupt pin.

The interrupt tests might not work on an XT-class computer if any printer port higher than LPT1 is used since the LPT2 interrupt might be used for a hard disk. Use caution when testing interrupts on an XT or with an early version of DOS.

I used a 6-foot (1.8m) 25-wire shielded cable to connect the computer to the Parallel Expander during tests, with a 3-foot (0.9m) 25-wire shielded cable serving as an input-to-output loopback cable. There were no failures or problems during many hours of testing with this setup. I can't overemphasize the importance of using good-quality shielded cable; make sure cable shields are connected to the connector shells.

TRADEOFFS AND APPLICATIONS

So, why should you use something like the Parallel Expander when lots of

special I/O boards are available, sometimes at very reasonable prices? In the case of a laptop, the answer is obvious-you use what you have. Actually, the thought of using a laptop as a tiny control console is quite an intriguing idea. In the case of larger PCs, the answer is not so simple. The Parallel Expander provides more I/O than the typical parallel I/O card, but a "special card" may operate faster or have more complete and immediately useful software. On the other hand, the Parallel Expander can be connected (without taking anything apart) to the many millions of existing PCs in just a few seconds.

Since the Parallel Expander doesn't do anything by itself, applications are up to you, but a quick glance at Figure **1** should cause quite a few ideas to spring to mind. For example:

*General-purpose I/O port
*Control and monitoring of single-board computers
*Connecting your PC to digital instruments

Listing **1**— *continued*

```
      TEST AL,$10
      JZ   @3
      SUB  AL,$08
@3:   OR   BX,AX
      ROR  BX,CL
      XOR  AX,AX
      MOV  DX,CTLPORT
      MOV  AL,$08
      OUT  DX,AL

      MOV  DX,CTLPORT        {Do nibb e 31
      MOV  AL,$09
      OUT  DX,AL
      MOV  AL,$0D
      OUT  DX,AL
      MOV  DX,INPORT
      IN   AL,DX
      AND  AL,$B8
      XOR  AL,$80
      SHR  AL,1;SHR AL,1;SHR AL,1
      TEST AL,$10
      JZ   @4
      SUB  AL,$08
@4:   OR   BX,AX
      ROR  BX,CL
      MOV  DX,CTLPORT
      MOV  AL,809
      OUT  DX,AL
      MOV  J,BX              {Save 16 bit word}
      END;
```

(continued)

Listing 1— *continued*

```
INW:=J:
END:

PROCEDURE OUTWORD(OW:WORD
BEGIN
    ASM
    MOV  BX,OW              Word in BX}
    MOV  AL,BL              Low byte in AL}
    MOV  DX,OUTPORT
    OUT  DX,AL              Send low byte to data lines}
    MOV  DX,CTLPORT
    MOV  AL,$02             Set up decoder to latch low byte}
    OUT  DX,AL
    MOV  AL,$06
    OUT  DX,AL              Do it}
    MOV  AL,$02
    OUT  DX.AL              Release decoder)

    MOV  AL,BH             {Send high byte in same way}
    MOV  DX,OUTPORT
    OUT  DX,AL
    MOV  DX,CTLPORT
    MOV  AL,$03
    OUT  DX,AL
    MOV  AL,$07
    OUT  DX,AL
    MOV  AL,$03
    OUT  DX,AL
    END;
END;
```

*Using optoisolators to monitor and control high-voltage systems

There must be hundreds more; as always, your imagination is the only limit to the potential applications for the Parallel Expander. ❑

John ***Lenihan has been a*** *Radio-* ***Electronic officer in the U.S. Merchant Marine for the last 15 years. Prior to that, he worked as an Electronic Technician and Field Engineer.***

SOFTWARE

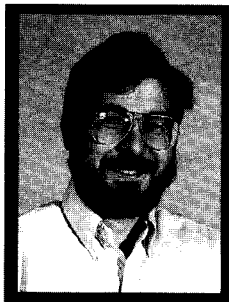
Software for this article is available from the Circuit Cellar BBS and on Software On Disk for this issue. Please see the end of "ConnecTime" in this issue for downloading and ordering information.

I R S

410 Very Useful
411 Moderately Useful
412 Not Useful

DEPARTMENTS

Absolute Power Corrupts: The '386SX Project Gets A Watchdog



Ed enters the hardware home stretch as he adds a single chip to handle watchdog functions, power monitoring, battery control, and RAM protection. The Firmware Development board is almost ready for action.

FIRMWARE FURNACE

Ed Nisley

"Power tends to corrupt and absolute power corrupts absolutely." While Lord Acton surely didn't consider firmware back in 1887, his epigram applies to our task this month. Putting vital data in RAM requires absolute control of both the power supply and the code...but preventing data corruption is what it's all about.

Adding a RAM chip to the Firmware Development Board is easy, but protecting it from harm is somewhat more difficult. You'll find the power monitoring and watchdog functions helpful even if your project doesn't need a battery-backed RAM.

PREPARE TO RAM

The new hardware this month has three main sections: the static RAM chip with its support chips, a Maxim MAX691 Microprocessor Supervisory Circuit, and a 16-bit I/O port. Refer back to the previous columns in this series for the complete schematics, as there's just not enough room to reprint everything every time!

Figure 1 shows the RAM and its support circuitry, which is quite similar to the (E)EPROM hardware earlier in this project. As promised, we now have firmware-controlled write protection with an LED to indicate when writes are enabled.

The Firmware Development Board's memory socket can hold RAM, EPROM, or EEPROM memory in either 8K or 32K byte sizes. The chips were designed with compatibility in mind, but Figure 2 shows the connections needed to adapt a single socket to

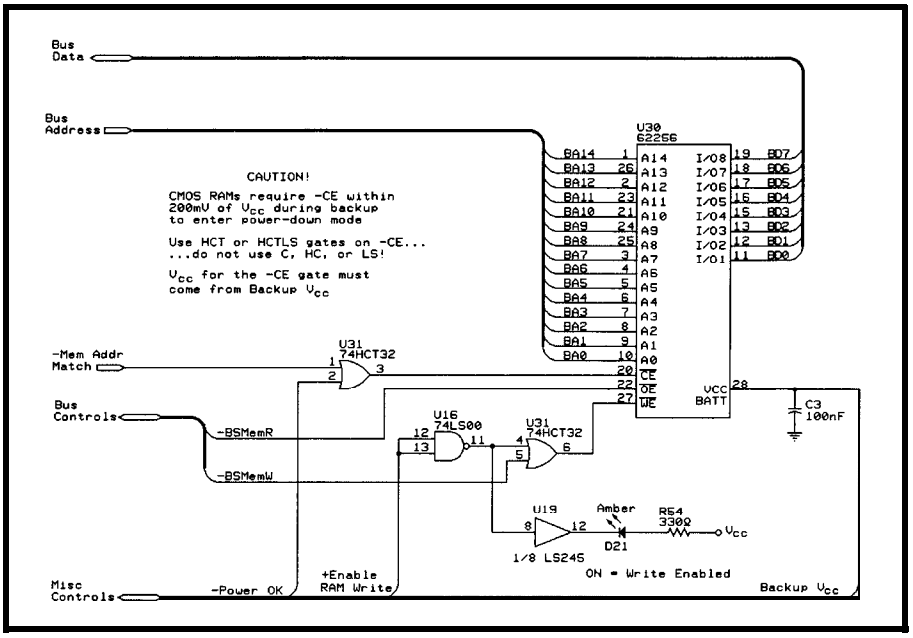


Figure I-Adding a RAM to the Firmware Development card uses circuitry similar to that presented last month. The two HCT32 gates control the Chip Enable and Wife Enable pins to prevent data loss during power loss. The -CE pin must be driven by a CMOS gate powered from the backup battery to ensure that the RAM enters standby mode.

the various pinouts. My board sprouted five jumper blocks to handle these options, but you won't need them if you pick just one chip and stick with it.

I used an Hitachi HM62256LP-150 RAM, but as you saw in last month's column, 8-bit ISA bus accesses allow more than 500 ns from the start of the –SMEMR or -SMEMW pulse. If you plan to use a backup battery, make sure the RAM is rated for low-power standby operation, which is typically shown by an "L" or "LP" part number suffix.

Unlike EPROMs and EEPROMs, static RAM chips require continuous power to maintain their data. Normal operation is specified at +5 volts, but they will retain data down to about 2.0 volts as long as you don't try to read or write it. Just reducing the supply voltage is not enough because the RAM still draws enough juice to drain a battery in short order.

Nearly all CMOS RAM chips nowadays feature a low-power standby mode which reduces their current consumption by several orders of magnitude. A chip that pulls more than 50 mA during a normal read may need only 10 nA in standby mode. Most of the chips enter standby mode when they are disabled, which is controlled by the Chip Enable voltage on pin 20.

But disabling the chip, even with a low supply voltage, is not enough. The data sheets specify the minimum -CE voltage to guarantee a maximum supply current. Because the supply voltage will vary depending on the battery condition, the voltage is actually specified as the difference between the voltages on pins 28 (the power supply) and pin 20. A 200-mV differential means that pin 20 is at 4.8 volts when pin 28 is at the normal +5 volts, but can be 2.8 volts when pin 28 is driven by a 3-volt lithium cell.

	Pin 1	Pin 20	Pin 26	Pin 27	Pin 28
8K RAM	n/c	Gated -CE	+CE (hi)	Gated –WR	Backup Vcc
32K RAM	At4	Gated -CE	AI3	Gated –WR	Backup Vcc
8K EPROM	V_{PP} (hi)	-CE	n/c	-Pgm (hi)	vcc
32K EPROM	V_{PP} (hi)	-CE	AI3	A14	vcc
8K EEPROM	-Busy (n/c)	-CE	n/c	Gated –WR	vcc
32K EEPROM	AI4	-CE	AI3	Gated –WR	vcc

The 8K EEPROM -Busy output on pin 1 must not be driven by external ciruitry.

Figure P-Although 8K and 32K byte RAMs, EPROMs, and EEPROMs all come in a 28-pin DIP package, there are some crucial differences. Five jumpers on the Firmware Development board cope with all the choices.

Figure 3 shows the result of a simple experiment measuring supply current as a function of -CE voltage. The vertical axis uses a logarithmic scale to compress the current, but it's easy to see when standby mode kicks in at about 4.5 volts. I ran the RAM at +5 volts, but the results are similar at 3 volts.

To ensure that -CE is held at the right level, you *must* drive it with a CMOS gate. Ordinary TTL gates cannot pull the input high enough, draw too much current for battery operation, and don't run at 3 volts anyway. The output from a CMOS gate is nearly at the supply voltage and will track the power supply as it switches to battery backup.

The spike at 1.3 volts exceeds 54 mA and occurs when the chip's internal logic passes through the range where both the p- and n-channel FETs conduct current. This is why you put lots of bypass capacitors on logic supplies and is where all the digital noise on your circuit board comes from.

The RAM's current draw has an exponential relation to chip temperature, so it may vary by nearly three orders of magnitude over the full temperature range. My graph represents room temperature, but I found that I could double the supply current by parking a desk lamp over the RAM chip. Pay close attention to the spec sheets when sizing the battery if you need extended temperature operation...those values are for real!

BACKUP WARNING

Although we've all seen and used the canonical diode-and-battery backup power circuit, there are good reasons to make things a bit more complex. I decided to use the venerable MAX691 because it has power monitoring, battery control, RAM protection, and a watchdog timer in a

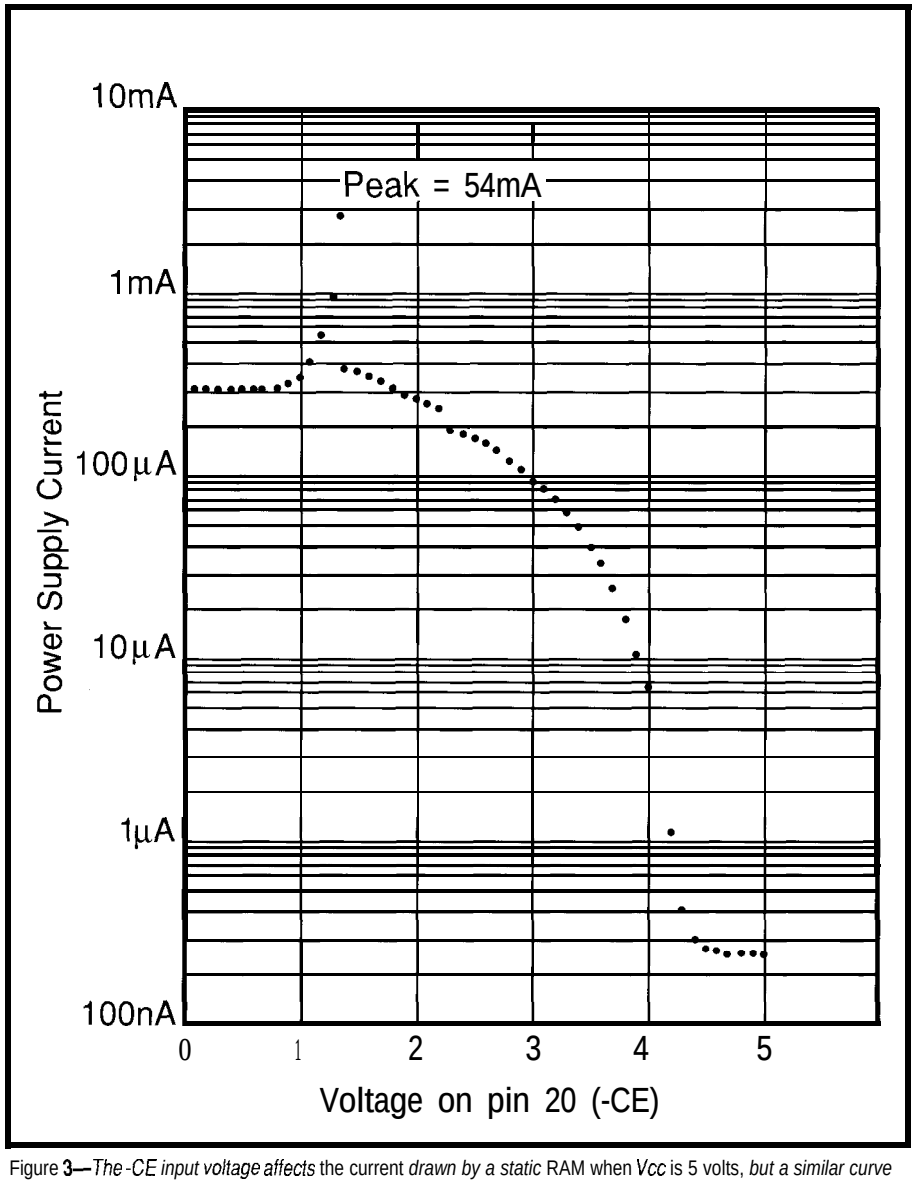


Figure 3—*The -CE input voltage affects* the current *drawn by a static* RAM when *Vcc* is 5 volts, *but a similar curve applies for 3-volt battery-backup operation. The –CE input must be within a few hundred millivolts of Vcc to put the RAM into standby mode. The 54-mA spike (which goes off the vertical scale) at 1.3 volts is caused by the chip's* infernal logic passing through the range where both the p- and n-channel *FETs* conduct current.

single IC. Other parts may be better for your particular application, but the '691 is a general-purpose workhorse.

Figure 4 shows the minimal external circuitry: most of the gates drive indicator LEDs that you might not need in a production system! I favor lots of LEDs to indicate what the firmware and hardware are up to, but, after all, this is a demo system.

Battery backup is straightforward because the MAX69 1 switches the voltage on pin 2 to the higher of the power supply on pin 3 or the battery on pin 1. I used a 3-volt, 250-mAH lithium cell, but any power source that provides enough voltage for the RAM will work.

An NEC Static RAM Application Note I reviewed for this project mentioned several UL requirements for lithium cell backup circuits. Even if your product doesn't need UL approval, the guidelines make sense. Bear in mind that I haven't read the UL regulations themselves, so don't depend on my suggestions to get your design approved!

Lithium cells react explosively to recharging, so you must prevent excess current from flowing into the cell. Typically, you would use a Schottky barrier diode in series with the battery because the forward drop of an ordinary silicon junction diode is far too high. The UL requirement limits the charging current to 1% of the cell's capacity, prorated by the possible charging time over the battery's service life. This can be a surprisingly small number, so check your diode specs carefully.

For example, if the power supply will be on 8 hours per day and the cell capacity is 250 mAH, the reverse charging current may not exceed 85 nA, which is derived from the following formula:

$$\frac{0.01 \times 250\ \text{mAH}}{8\ \text{hours/day} \times 365\ \text{days/year} \times 10\ \text{years}}$$

or about 85 nA.

The worst case is for continuously powered systems because the cell will always see recharging current.

You may need a bigger battery than the RAM's standby current would lead you to expect, if only to boost the allowable reverse charging current to a reasonable value.

The MAX691 limits charging current to 10 nA typical, 100 nA maximum, and 1µA over the full temperature range. This may not be good enough for a UL rating, particularly for extended temperature applications, so you may need a series diode anyway. I decided to skip the issue, as the Firmware Development Board is not intended to be UL rated!

The UL requirements specify a current-limiting resistor in case the diode is damaged. The fault current is 5 mA regardless of battery capacity. The resistor value is the maximum possible supply voltage minus the cell voltage divided by 5 mA, which works out as follows:

$$\frac{(5.5-3)\ \text{volts}}{5\ \text{mA}} = 500\ \text{ohms}$$

The next higher standard value is 560 ohms. I included this resistor to prevent problems should the MAX691 succumb to a static zap, but I'll admit this isn't consistent.

The MAX69 1 requires a bypass capacitor on pin 2 to stabilize the internal voltage comparator and switch. It's also essential because the chip can supply only 50 mA of current even when powered from the normal supply. If your circuitry requires more than that, the data sheet shows how to

boost the current without affecting the backup battery.

The bypass cap must store enough energy to stabilize the voltage during the huge current spike shown in Figure 3. You should also bypass the static RAM at its socket, as transient currents are offended by long wires.

With power assured, the next step is controlling the CPU during the switch over. After all, it does no good to preserve data scrambled by a power-starved processor!

DATADEFENSE

The Original IBM PC power supply produced a "Power Good" signal that held the CPU in reset until all of the power supply voltages were stable. When you flipped the Big Red Switch, the Power Good signal dropped before the supply voltages failed. In effect, the system always saw clean power when it was running.

The ISA bus RstDrv (Reset Drivers) signal is activated whenever the system board sees a hardware reset. In principle, this line should be activated whenever the Power Good signal is low so that all of the PC's circuitry is reset while the power is out of tolerance.

However, to quote Solari, "The above information...is a combination of...the IEEE P996 specification and various IBM technical reference manuals. It is sometimes unclear which platforms adhere to these specifications."

I've seen supplies without a Power Good signal, evidently depending on the system board's (nonexistent) reset circuitry. In fact, one group I worked with simply tied the system board's Power Good line to a capacitor and ignored the fact that "Power Good" was active long after the power went bad. I argued in vain for a power monitor chip, but the board was already laid out and it was easier to kludge the cap than add an IC.

The MAX691 monitors the supply voltage on pin 3 and triggers several actions when it falls below specific levels. While these may not be strictly necessary in a PC with a good power supply, as long as we're using the chip we may as well put it to good use. If,

Listing 1--Producing a regular heartbeat on the wafchdog pin requires an interrupt handler attached to a timer tick. This code rotates a 16-bit variable and sends the high-order bit to the watchdog. To avoid sending the bits faster than the eye can follow, if counts interrupts and sends one bit every WATCH_RATE ticks. The mainline code must reset the WatchPending flag at least once every 16 bits to prevent this code from forcing a watchdog reset.

```
HandlerWD() {

asm {
*
          PUSH  AX                  save bystanders
          PUSH  DX
          PUSH  DS
          MOV   AX,CS               aim at our segment again
          MOV   DS,AX
*
* Count down the interrupts until we need a watchdog update
*
          DEC   <WatchDivide
          JNZ   WD_Ret
          MOV   <WatchDivide,#WATCH_RATE
*
* Decide if a new watchdog word is needed
* If it is,  and the mainline code is jammed, we lock up and die
*
WD_Tick   DEC   >WatchCounter
          JNZ   WD_Go               nonzero says use old bits
          CMP   >WatchPending,#0    has mainline code reloaded bits?
          JE    WD_Load             zero says yes,  so we are golden
          MOV   DX,#LED_ADDR_A      nonzero says we have trouble
          MOV   AX,#~$8000          left decimal point flags problem
          OUT   DX,AX
*
WD_Lock   JMP   <WD_Lock            stay here until watchdog timeout
*
WD_Load   MOV   AX,WatchBits        fetch new bits
          MOV   WatchShift,AX       . . . for the shift reg
          MOV >WatchCounter,#16 reload the counter
          INC   >WatchPending       set flag for mainline code
*
* Blip the watchdog output to ensure a transition every time
*
WD_Go     MOV   DX,#CTLS_ADDR_A     set up for watchdog output
          MOV   AX,CtlsCopy         get existing bits
          AND   AX,#~WATCHDOG_A     send a low (LED ON)
          OUT   DX,AX
          Punt
          OR    AX,#WATCHDOG_A      send a high (LED OFF)
          OUT   DX,AX
*
* Rotate the watchdog bits and send the high one
* We flip the bit so 1 turns the LED ON like it should
*
          ROL   >WatchShift,1       get high-order bit in C
          JNC   WD_Z                clear says leave the output high
          AND   AX,#~WATCHDOG_A     set says make output low
WD_Z      OUT   DX,AX               send it out
          XOR   AX,#WATCHDOG_A      flip the bit back again
          MOV   CtlsCopy,AX         save for next time
*
WD_Ret    POP   DS                  restore bystanders
          POP   DX
          POP   AX
          POP   BP
          IRET                      restore stacked flags
}

}
```

for whatever reason, you are *not* using a standard PC supply, this circuit will ensure that the RAM's contents are intact regardless of what happens to the rest of the system.

Recall that we must put the RAM into standby mode when the power fails. The MAX69 l's -CE Out signal tracks -CE In until the supply voltage falls below 4.65 volts, at which point the MAX691 forces -CE Out high. This disables the RAM and puts it into standby mode.

Unfortunately, while the MAX691's nominal delay is 50 ns from -CE In to -CE Out, the maximum is 200 ns. That's OK for this relatively slow ISA bus application, but I felt I should show how to adapt it to faster systems. Maxim obviously took some hits on this, as they now have a MAX691A with a far more useful 10-ns nominal delay.

The key is to control a faster logic gate with a "DC" signal. As shown in Figures 1 and 4, if -CE In is grounded, the HCT32 gate delays the RAM chip select by only about 20 ns. When -CE Out goes high, the RAM is in standby mode with its -CE pin driven nearly to the supply voltage by the CMOS gate.

Obviously the external gate must be powered by the backup battery through the MAX69 1. You should use an HCT gate rather than C or HC to ensure that the inputs respond to TTL switching levels. Pure CMOS gates have V_{IH} thresholds well above the normal TTL V_{OH} level and may not work correctly when driven by TTL gates.

PROCESSOR PROTECTION

Although the data in RAM is now safe from harm, It would Be Nice if the CPU knew what was going on too. After all, simply disabling the RAM may cause invalid data if the CPU was in the midst of a multibyte update. Although the power may be failing, a millisecond gives you a lot of time to put things in order.

The MAX69 1 can provide an early warning of impending doom by monitoring the voltage on its Power Fail Input pin: when that voltage drops below 1.3 volts, the Power Fail Output pin goes high. The resistor divider and trimpot shown in Figure 4 set the trip point so that –PFO is active before the RAM is disabled. You can set the voltage without a trimpot, but this lets you activate –PFO and test the system without blipping the supply voltage.

Although you could wire –PFO through an inverting driver to one of the system's interrupt lines, if interrupts are masked off when the power fails, all is lost. The solution is to use the –IOCHCK (IO Channel Check) ISA bus line, which activates the CPU's NMI (Non-Maskable Interrupt) pin. That interrupt cannot be ignored, so the interrupt handler is sure to get attention.

Once the NMI handler is in control, it can take whatever steps are needed to ensure a safe and orderly system shutdown. With only a few milliseconds of power left, however,

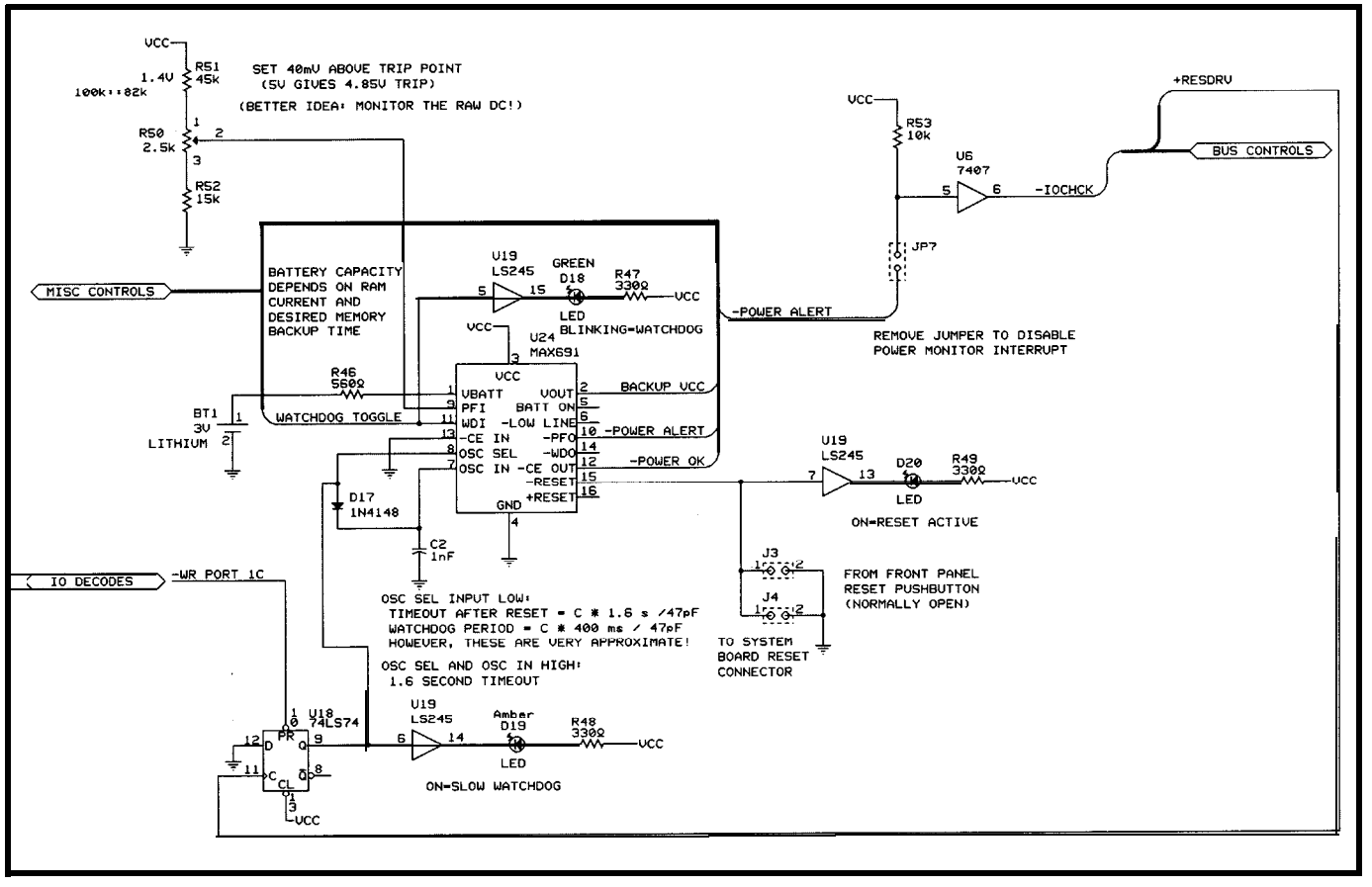


Figure 4—*The MAX691 monitors* the power supply, warns *of impending* power *failure,* controls the backup *battery and -CE switching, and includes a variable-rate watchdog* timer. *The LS74* flip-flop ensures that the watchdog *times out after about* 30 seconds following a hardware *reset;* any *access to port 31C* reduces *the* timeout to 1.6 seconds. *Much of the remaining circuitry drives indicator LEDs to reveal what's going on.*

saving data to disk, sending a message out the serial port, or doing anything on a human scale just won't work. Think fast and think final!

The MAX691 activates its -Reset output when the supply voltage drops below 4.65 volts. In a good PC, the spec for the +5 V power at the card connectors is 4.875 V minimum, so the Power Good signal should occur before the MAX691 triggers a reset. The MAX691 also has a +Reset output which you can use directly on 8031 systems. Two additional power monitor outputs, Battery On and Low Line, are useful in some systems. Check the data sheet for further hints and tips.

To recap, the sequence of events during a power failure starts with –PFO activating the CPU's NMI input. The interrupt handler prepares for the coming shutdown and then enters a loop until either the MAX69 1 or the Power Good circuitry detects an invalid voltage and activates the system reset. The MAX691 disables the RAM at the same time it activates the reset line.

When power comes back on, Power Good and the MAX691 decide when the voltages are within tolerance and release the system reset line. The BIOS then gets control and the system boots normally. The RAM is enabled when the MAX691 releases the reset line, so the RAM will be ready for the first firmware access.

Figure 4 shows connections to both RstDrv and the system board Reset connector. The two are not identical: Reset is normally wired to the front-panel Reset switch, while RstDrv is an ISA bus output. You cannot drive RstDrv and you do not have direct access to the signal that actually resets the CPU.

I kludged a small adapter from jumpers and header pins for the Reset connection: the front panel switch plugs into the adapter, which then plugs into the system board. A two-conductor wire joins the adapter to a header on the Firmware Development board. If you connect the thing backwards, the FDB's ground will hold system reset low, but that goof is easy to find.

Listing P--This *routine decides if a Non-M&able Interrupt was caused by the MAX691's Power Fail defector. If so, if write-profecfs the RAM, lights a decimal point, and enters a spin loop waiting for Reset. If not, if passes control to the NMI handler set up by the BIOS.*

```
HandlerNMI() {

asm {

            PUSH  AX                  save bystanders
            PUSH  DX
            PUSH  DS
            MOV   AX,CS               aim at our segment again
            MOV   DS,AX

* Check to see if the power fail bi t is active

            MOV   DX,#STAT_ADDR_A
            IN    AX,DX
            Punt
            TEST  AX,#PWR_GOOD_A
            JNZ   NMI_Chain           nonzero says not our problem

* We have a power failure, so write-protect the RAM and lock up

            MOV   DX,#CTLS_ADDR_A     turn off write-enable bit
            MOV   AX,#~NV WENABLE_A
            OUT   DX,AX
            Punt
*
            MOV   DX,#LED_ADDR_A      show that we are locking up
            MOV   AX,#~$0080          with right decimal point ON
            OUT   DX,AX
            Punt
*
NMI_Lock    JMP   <NMI_Lock           jam up here until next reset
*
*
* Chain to previous NMI handler
*
NMI_Chain   POP   DS                  restore bystanders
            POP   DX
            POP   AX
            POP   BP
            JMP   CS:>Int020ff        indirect to old handler
}
}
```

FIRMWARE SUPERVISION

The MAX69 1 has one additional feature that I believe is essential for any embedded system: a watchdog timer. As any INK reader should know, a watchdog is simply a timer that resets the system after a predetermined interval after a transition on its input pin. The firmware must wiggle that bit often enough to prevent the timer from timing out.

The principle is simple: correctly functioning firmware will reset the timer, while locked-up or stalled code will not. A system reset clears the slate and starts all over again; presumably whatever the system is controlling can stand a glitch in the outputs while the CPU recovers its wits. If your system can't stand a brief interruption a watchdog isn't for you...but you must provide some other way to detect failures and lockups, because they will occur!

A particular problem with embedding a stock PC is that the BIOS gets control when the CPU reset signal goes inactive and holds it until the disk boot is finished. As a result, just after reset the watchdog must allow about 20 seconds for the system's normal boot process. But a 20-second timeout is probably far longer than you're willing to wait when your

firmware should be in control, so we need a variable-rate watchdog.

I've seen some systems that allow you to disable the watchdog, but I don't like that because a firmware fault or hardware glitch can (nay, will!) find that chunk of code and disable the watchdog just before taking a permanent walk in the woods. A variable-rate watchdog ensures that the reset will occur eventually.

Figure 4 shows how I adapted the MAX691's watchdog. The LS74 flip-flop is cleared by the ISA bus RstDrv signal. When the MAX691's Osc Sel input is low, its watchdog runs at a frequency set by the external capacitor. In this case, the 1-nF cap sets a watchdog timeout of about 30 seconds, which is long enough to load and start a program from disk.

The first time the code writes to port 3 1 C on the Firmware Development Board it sets the flip-flop, which raises both Osc Sel and, through the diode, Osc In. When those inputs are high, the MAX691 runs from an internal oscillator that causes a time out after 1.6 seconds, which is fast enough for normal operations.

The MAX691 data sheet has formulas to compute the external capacitor value for a given timeout, but I've found that they are not particularly accurate. You may need to experiment to find the right value for your application. Remember that a slow watchdog is better than a fast one in most cases!

Figure 5 shows the new I/O bits on port 3 1 C, which is identical to port 31E that we used for the LED digits and DIP switches. Although only three bits are defined thus far, I've got plans for the remainder-never fear!

The Firmware Development Board now sports several indicator LEDs so you can tell at a glance when RAM writes are enabled, Reset is active, the watchdog is toggling, and how long a watchdog timeout will take. The LED drivers are part of the LS245 I used for the interrupts from the 8254 timer, so the outputs are always enabled.

GETTING DOWN TO CODE

The RAM is similar enough to the (E)EPROM we covered last month that I just converted MEMTEST test program into RAMTEST by ripping out the EEPROM write timing and expanding the memory tests to include all 32K bytes. There's nothing new here, so I won't show the listings, but do download the code to check out your wiring.

Although a watchdog timer is essential for a production system, it can be a serious nuisance while you're developing and testing code like RAMTEST. I disabled my board's watchdog by yanking the system board Reset connection. The red LED then indicates when the MAX691's Reset output is active, which helps track down problems: if it ever goes on, you've goofed!

But you do need some way to verify that the watchdog and power monitor code is working, so I wrote DOGTEST. Because the watchdog is active, you must boot DOGTEST from diskette so it gets control before the initial 30-second timeout expires; it then sets up the interrupt vectors and begins toggling the watchdog output.

The watchdog doesn't care how often you toggle its input bit as long as you do it often enough. If, however, there's an LED on that bit, it is a Very Good Idea to produce a regular "heartbeat." There is something unsettling about an irregular LED even if it does indicate perfectly good code.

I use heartbeat LEDs as output devices: a regular blink signifies normal operation, while long and short blinks report errors. The code is actually pretty straightforward: a timer interrupt handler takes care of timing, while the mainline code sets up the bit patterns. I've used this trick on many systems, so you can probably adapt it to yours.

Listing 1 shows DOGTEST's timer interrupt handler. The mainline code attaches this function to Int 1Ch, which the BIOS invokes after every 54.9-ms timer tick. I divided that down to 6 bits per second, so the interrupt handler runs through the WatchBits variable in about 2.6 seconds.

The interrupt handler sets WatchFlag when it finishes sending all 16 bits. If WatchFlag is still set

after 16 more bits, the interrupt handler enters the tight loop at WD_Lock. Because the watchdog output bit no longer toggles, the MAX691 will eventually reset the system.

The mainline code thus has two responsibilities: it must load a bit pattern into WatchBits at least once and it must clear WatchFlag at least every 16 bit times to prevent a time-out. This means the maximum delay until a reset is 2.6 seconds to shift all the bits out plus 1.6 seconds after the last bit, or 4.2 seconds overall.

The most soothing bit pattern is FFO0, which produces a reassuring heartbeat with I.3 seconds on and 1.3 seconds off. AAAA produces an exciting 3-Hz blink, while F140 sends a "one long, two shorts" blink code that might indicate a particular failure or error condition. YOU can do a surprising amount with 16 bits if you have to!

Note that 0000 is a perfectly valid, albeit dull, pattern that will *not* cause a watchdog timeout. The interrupt handler forces a transition between each pair of bits, so the watchdog sees a pulse every 165 ms regardless of the bit values. If you look closely at the LED in a dark room you can see those 1.3-µs pulses. Try it!

DOGTEST's main loop is quite simple: it checks and resets WatchPending so the interrupt handler remains happy, copies the DIP switches into WatchBits so you can experiment with different bit patterns, and writes a counter value into the LED digits so you can see something happening.

Bit	Function
7	1 = System board parity check
6	1 = IO channel check
5	1 = Timer 2 output bit
4	Toggles with each RAM refresh
3	0 = IO channel check enabled
2	0 = System board parity check enable
1	1 = Speaker data enabled
0	1 = Gate Timer 2 output to speaker

Figure 6-A *Non-Maskable* Interrupt can *be caused by a system board parity check or fhe ISA bus –IOCHCK signal. Your firmware can determine* which *input is active and mask if off by using these bits in I/O port 0x61. Some systems have additional NMI sources with differenf controls. Bit 7 in port 0x70 must a/so be zero to enable fhe CPU's NMI input.*

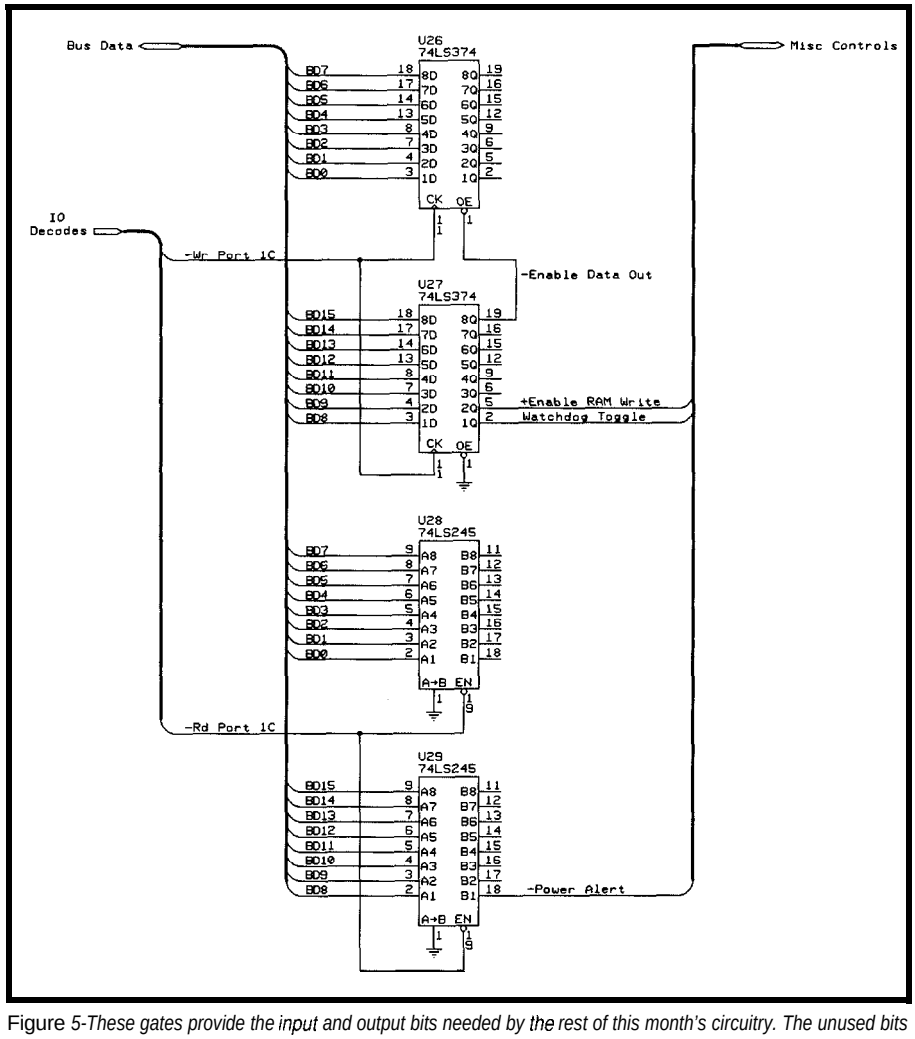


Figure *5-These gates provide the input and output bits needed by the rest of this month's circuitry. The unused bits will come in handy for future projects!*

DOGTEST also accepts a command from the serial port: if you press the "1" key, it will stop clearing WatchPending to force a watchdog reset. The interrupt handler turns on the left decimal point just before it enters the final loop and the MAX691 should activate Reset about 1.6 seconds later.

UNMASKING THE NMI

By definition the CPU cannot ignore a Non-Maskable Interrupt. However, the IBM PC and its descendants include circuitry to prevent a signal from reaching the CPU's NMI pin. While this may seem contradictory, the system may not be able to start, let alone operate correctly, with a hot NMI.

For example, if an NMI occurs before the firmware validates RAM, loads the stack pointer, and sets the NMI vector, the system will crash. The CPU can accept an NMI immediately after its Reset input goes inactive, so if NMI is stuck active, the CPU cannot even run diagnostics to pinpoint the problem.

However, it's *not* a good idea to leave NMI off all the time, so IBM's AT engineers picked a distressingly clever way to control it. The MC1468 18A Real-Time Clock has 64 bytes of nonvolatile RAM addressed by the value written to I/O port 70. The clock ignores the two high-order bits, so the engineers added a latch to bit 7 that inhibits NMI: simply write address 80 instead of 00 to mask the unmaskable.

Wish you'd thought of something like that for your last project?

The latch holds the mask bit and there is additional circuitry to turn it on during a hardware reset. It remains set until the BIOS writes an RTC address between 00 and 7F, which

happens only after the BIOS is sure everything is ready. Thus, a hot NMI won't disrupt normal system diagnostics.

NMI can be activated by a variety of sources depending on exactly which AT or clone you have. The two standard sources are the system board parity check hardware and the –IOCHCK signal from the ISA bus. These signals are controlled by bits in I/O port 61, as shown in Figure 6.

DOGTEST's NMI handler, shown in Listing 2, is much like the interrupt handlers you've seen before, with one key exception. Because the NMI does not pass through the external 8259 interrupt controller chips, the handler must not send out an EOI in response to the interrupt.

The code examines the MAX69 l's –PFO bit through port 3 1 C; if it's zero, a power failure is impending. Otherwise, the code simply invokes the previous handler set up by the BIOS during the power-on sequence.

Because further interrupts are blocked out until the CPU executes an I RET instruction, the tight loop at NM I_Loc k could be replaced with a H 1 t I favor a loop so I can add a few instructions to toggle an output bit that flags the event on a scope, but the choice is yours.

RELEASE NOTES

The code on the BBS this month includes C and BIN files for RAMT E ST and DOGTEST. Remember to boot DOGTEST directly from diskette so it gets control before the MAX691 resets the system.

I've also tweaked the LOADEXT. ASM routines from last month. You can now load a BIOS extension from diskette into either EEPROM or RAM and set the checksum on the fly.

OK, that's enough hardware! If you can't start doing embedded PC code with what we've got now, it's time to dust off your COBOL manuals. Next month, I plan to spend some time exploring BIOS extensions, hardware and firmware resets, and the worst hack in PC-dom.

Ed Nisley, as Nisley Micro Engineering, makes small computers do amazing things. He's also a member of the Computer Applications Journal's engineering staff. You may reach him on CompuServe at 74065,1363 or through the Circuit Cellar BBS.

SOURCE

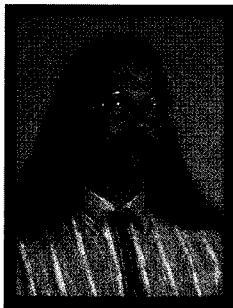
Pure Unobtainium has the MAX691 and selected parts for the Embedded '386SX series, as well as the schematics for everything to date. Write for a catalog.

Pure Unobtainium
13 109 Old Creedmoor Rd.
Raleigh, NC 27613
Voice/Fax (9 19) 676-4525

I R S

413 Very Useful
414 Moderately Useful
415 Not Useful

Take a Swipe at Optical ID Cards



Stuck on how to come up with a data input system for identification purposes or logging data? Using off-the-shelf components and your PC's joystick port, Jeff builds an innovative bar code reader.

FROM THE BENCH

Jeff Bachiochi

Can you guess how many different cleaning agents are for sale at the average supermarket? Today I counted 35 brands. Of those, 25 were available in a pump and 10 in an aerosol. They range in color from the deepest blue to the most fluorescent orange. Most will clean blueberry stains without scratching your precious porcelain surfaces.

Manufacturers seem to spend more money on packaging and advertising than on the actual product. Products today aren't good enough if they just clean. They must also kill bacteria, be gentle, and leave a pleasant scent behind. But none of these products can eliminate the unpleasant task of data entry.

SWIPE (TO THE RESCUE)

Supervisors and employees agree: task management has never been much fun. However, it is critical to the delicate balance between profit and loss for the company. Most companies seem to drown in a sea of paperwork. Some of this recordkeeping can be simplified and automated if the correct tools can be found.

Bar code wands have taken us a step closer toward automated data entry. The wand is usually tethered by an umbilical cord which carries both power to and data from its light-sensitive tip. Data is presented as reflective/nonreflective areas to the wand's infrared transmitter/receiver, converting the patterns into digital data signals.

Every time you use an ATM, your card's magnetic data is converted into digital data by a magnetic read head. Besides the obvious difference between media, there is a secondary difference. Bar code readers are brought to the data while card readers have the data brought to them.

I will often use the ATM even during "banking hours" rather than stand in the queue. As far as machines go, it is one of the most user friendly around, and after all, if you do make it up to a teller, they will call your account up on the computer anyway. So, I avoid the middleman and speak directly to the source.

This month, I combine these two data collection methods to produce an inexpensive and easily implemented data input system. In its simplest form, it could be used as an identification device or to keep a complete log

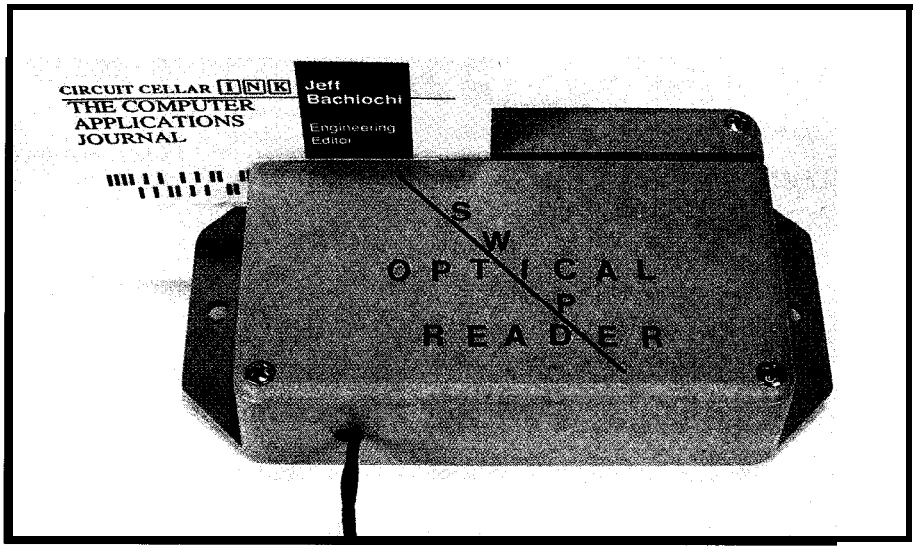


Photo 1-I *The Swipe Optical Reader is fairly easy to build and serves well as a portable unit*

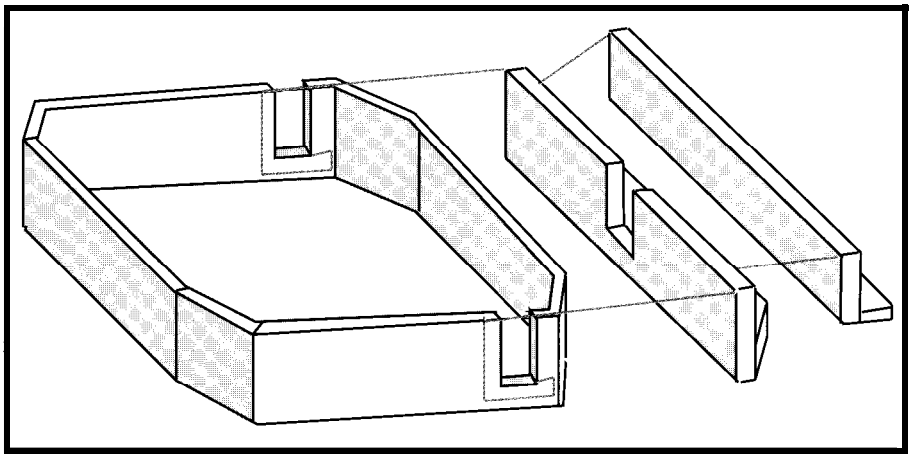


Figure 1—*Two pieces of scrap plastic (rails) are placed in line with the enclosure tabs. A card is "swiped" through these tabs in between the two rails. A photosensor positioned at the left rail slot reads data as it passes by.*

for task management or security purposes.

The heart (better yet, eyes) of this month's project uses a pair of reflective photomicrosensors stacked inside a small 3" x 4" enclosure. The enclosure is modified with a card slot and voilà: an optical swipe reader is born. Two sensors are used to provide two tracks of information. This configuration opens many possibilities for experimentation.

ENCLOSURE PREPARATION

Since the enclosure I have chosen has mounting tabs on the bottom, I slotted the top surface, parallel to the longest dimension. I adjusted my table saw blade for a depth of $^1/_2$", set the rail at 1", and ran the enclosure through top side down. Always use a feed stick to move your work through the business end of the table saw; you'll probably need those fingers later.

The slot supports are made from scrap pieces of plastic, although you might want to use extruded aluminum angle. A single right-angle piece forms one side and the bottom of the slot. This is glued in place at the appropriate level even with the bottom of the slot. A second piece sits on the first. A small spring keeps the second piece pressed loosely against the first. When a card is inserted between the first and second piece, the spring's tension holds the card against the guide at the appropriate distance from the sensor. Figure 1 shows how the card guide is assembled.

SENSOR SELECTION

The reflective photomicrosensor system uses an infrared light source and a phototransistor (diode) to pick up the reflected light energy. These devices are available separately or packaged together as a photosensor. Photosensor housings aim the light source and sensor such that they converge at a predetermined distance or focal length. The reflective surface should be placed at this distance for maximum sensitivity. Two such photosensors, available from Digi-Key, are the EE-SY101 and EE-SY148, both made by Omron. The '101 is a TO-92–sized device with a focal length of 1 mm. I mounted these along the edge of a small piece of protoboard. Refer to Figure 2 for the circuit I used to support these photomicrosensors. The comparator has an adjustable trip point (POT1) and hysteresis (POT2). The output of the circuit is forwarded to a four-pin connector that provides connection points for both power and the conditioned sensor outputs.

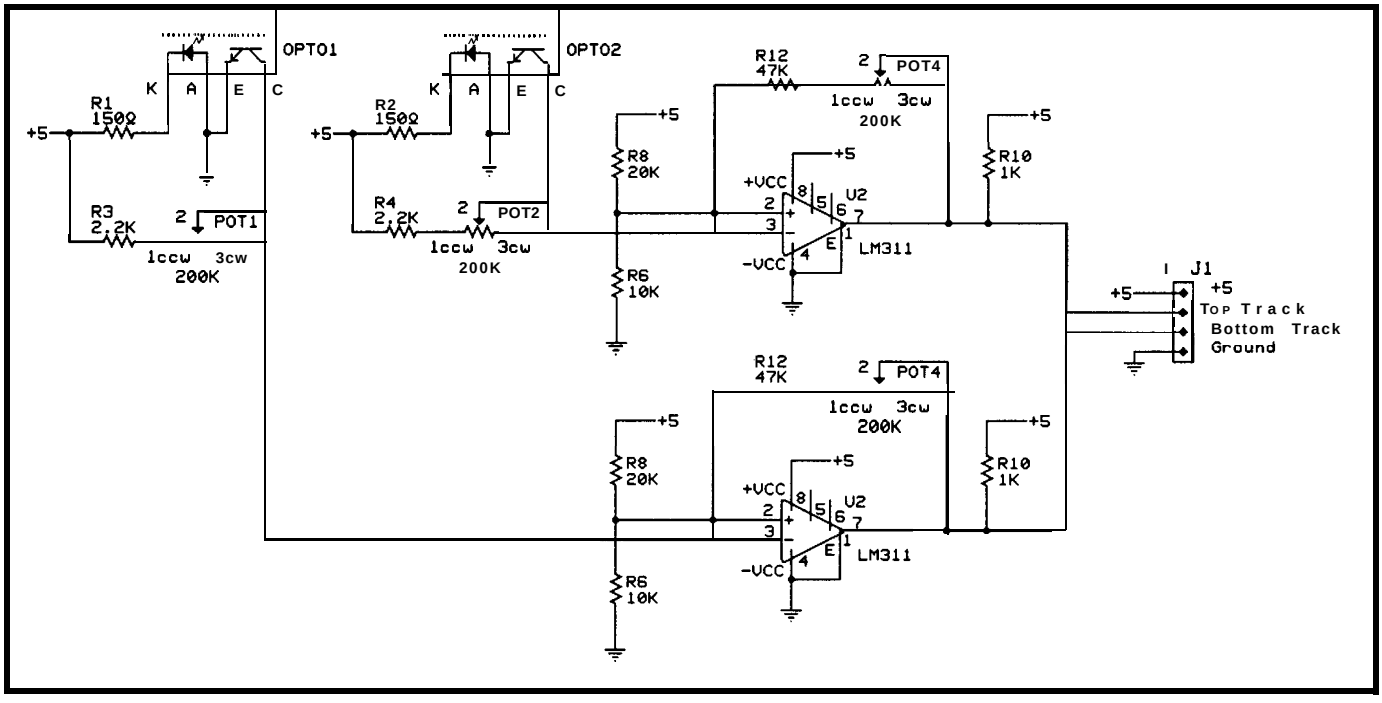


Figure 2-A *photosensor package (EE-SY101) consists* of a *reflective photomicrosensor, which uses an infrared light source and a phototransistor to pick up the reflected light energy.*

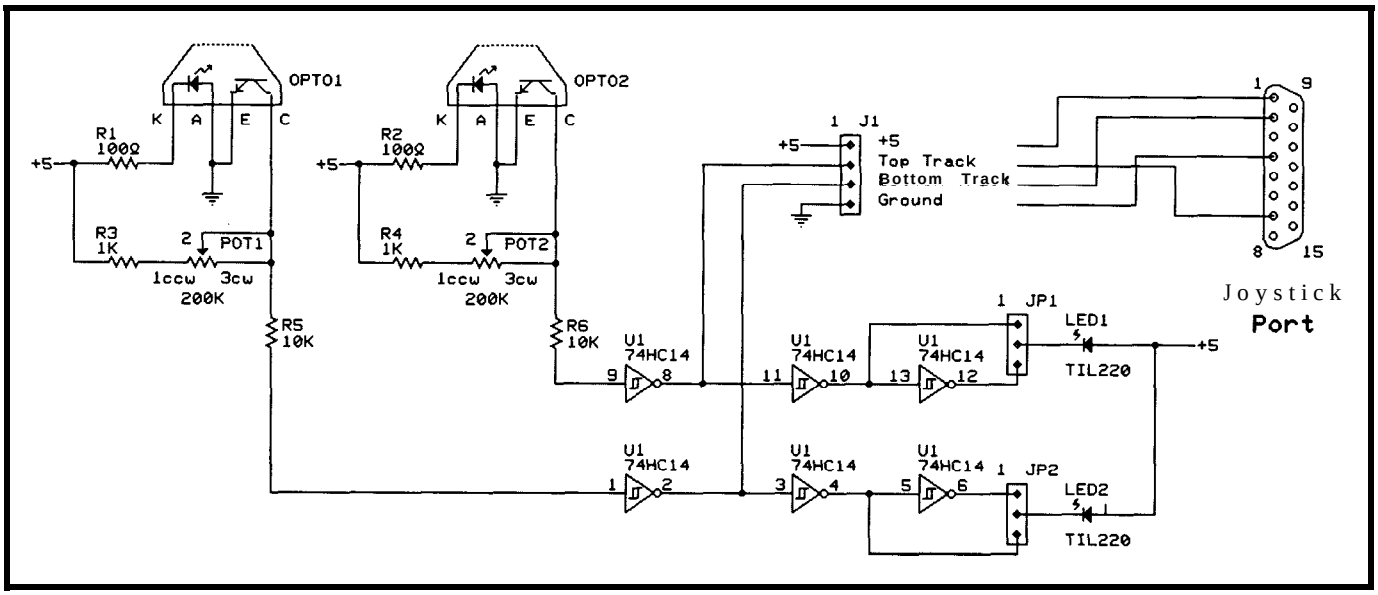


Figure 3—*Another, larger photosensor package (EE-SY148) uses a 74HC14 for circuit hysteresis. In addition, focal length adjustments are easier because the mounting hole between the transmitter and receiver is elongated.*

The second sensor, the '148, is a larger package. This wedge-shaped device has a focal length of 3 mm. This time I used a 74HCT14 to give the circuit a little hysteresis; see Figure 3 for the circuit I used with this device. Mechanical support and alignment is easier with these devices because they have an elongated mounting hole between the transmitter and receiver that makes focal length adjustments more manageable. I wired an output connector with the same configuration as with the previous circuit to allow the sensor circuits to be easily exchanged within the enclosure.

SIMPLE INTERFACE

I've used the PC's parallel port many times for interface projects. However, this time there is an advantage to using a different port. Since we're dealing with a device that provides input signals only, the PC's joystick port has all the necessary signals needed to support this circuit. It can provide power since it has +5 volts and ground normally used for the joystick's potentiometer, and it has push-button inputs that are pulled high internally with 1 k resistors and grounded by pushing a button. Using BASIC, the status of each push button can be polled to determine whether the attached sensor is seeing reflected light or not.

Standard bar code techniques encode data as line width and/or spacing widths. This method is sensitive to constant scanning speed in order to accurately determine relative line/space widths. You may wish to experiment with this method, but since I have two tracks available, I can use a simpler approach.

No matter what approach you choose, there is a need to determine where the actual data starts and in what sequence (from what direction] the data is being entered. Therefore, a start flag and an end flag should be used to frame the data sequence. You can see the standard I settled on for my setup in Figure 4.

False sensing can occur when the card enters the sensor's detection zone. So, by using a minimum of three marks on one track followed by a space, a start code is recognized. False codes can occur prior to this without affecting the recognition of a true start. If the opposite sequence is used as an end code, the direction of the swipe can be established. This can only safely be assumed if you know how many data bits are between the start

```
                    possible false code
                          start code          end code
                              data
                    :     :   :                 .
data track          1     1110xxxxxxxxxxxxxxx1000
                          :   :                       or
                          :   :                 .
data track          0     1110xxxxxxxxxxxxxxx1000
```

Figure 4—*Start and end codes must be used in any encoding scheme to frame the data and to reject false readings.*

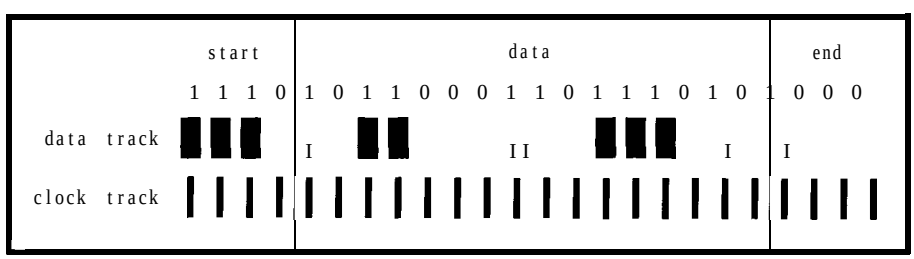


Figure 5—*In the simplest data format, the upper track is used for data while the lower track contains clock pulses.*

and end codes (especially since the data may contain a sequence that looks like a start or end flag), or you take the complement of each data bit, in which case three sequential marks or spaces are not legal.

Figure 5 shows the simplest data format using the lower track for the clock and the upper track for the data. In this format, the top track is searched for data when the bottom track loses signal (hits a nonreflective black mark). To keep the bidirectional benefit of the swipe input, the format of clock to data width is 1:3. The data must extend beyond both ends of the clock mark to assure legal data recognition independent of which direction a clock edge is encountered. This also increases the need for perfect alignment. Figure 6 illustrates this technique of data encoding for "1" and "0" data bits.

I used the code in Listing la to print clocked bar codes on my HP LaserJet Series II. Run the program in Listing lb to poll the PC's joystick port and display the received data-bit sequences. If the start code, data, and end code are received as expected, a beep declares an accepted swipe. Bit errors are displayed as "." and timeouts as a "–".

Figure 7 shows an alternate format that uses the bottom track as data "0" bits and the upper track as "1" bits. In this format, both tracks are watched and data is assembled as the marks are reached in a self-clocking format. Unlike the previous clocked format, this requires fewer character spaces per bit (we're dealing with edges now).

DATA INTEGRITY

The fact that data of a fixed length is surrounded by proper start and end codes ensures data integrity to a high degree. Additional steps can be taken to increase data integrity. You might want to add a simple CRC integrity bit or complement every bit of data. The tradeoff here is the maximum number of character places which will fit on a card.

I've posted code on the BBS similar to that in Listing la to print a self-clocking format that uses complemented data bits to assure high

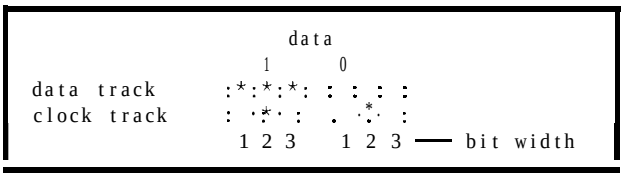


Figure 6—*The width of data marks must be three times as wide as the clock marks to assure reliable scanning in either direction*

Listing 1 a—*Very low density bar codes can be printed using standard IBM character graphics.*

```
10 REM LOWER TRACK IS CLOCK, UPPER IS DATA
20 H$=CHR$(219)+CHR$(221)                    :REM DATA MARK CHARACTER
30 L$=CHR$(32)+CHR$(32)                      :REM DATA SPACE CHARACTER
40 C$=CHR$(222)+CHR$(32)                     :REM CLOCK CHARACTER
50 FOR C=1 TO 24                             :REM BUILD A CLOCK TRACK
60 CLK$=CLK$+C$
70 NEXT C
80 INPUT"Enter a number (0-65535)?",X
90 X(0)=INT(X/256):X(1)=X-(X(0)*256)         :REM BREAK IT IN TWO
100 S$=""                                    :REM INITIALIZE DATA TRACK
110 S$=S$+H$+H$+H$+L$                        :REM ADD START CODE
120 FOR B=0 TO 1                             :REM BOTH BYTES
130  FOR Z=7 TO 0 STEP -1                    :REM  MSbit TO LSbit
140  IF (X(B) AND (2^Z))=2^Z THEN S$=S$+H$ ELSE S$=S$+L$
150 NEXT Z                                   :REM DO ALL BITS
160 NEXT B                                   :REM DO BOTH BYTES
170 S$=S$+H$+L$+L$+L$                        :REM ADD END CODE
180 LPRINT S$                                :REM TOP TRACK
190 LPRINT CLK$                              :REM BOTTOM TRACK
200 LPRINT
210 GOTO 80
```

Listing 1b—*Data is received at a low enough rate that BASIC can poll the joystick port and decode the incoming steam.*

```
10 STRIG(0) ON              :REM ENABLE JOYSTICK BUTTON
20 STRIG(4) ON
30 T=0                      :REM TIMEOUT FLAG = NONE
40 S$="":T$=""              :REM CLEAR STRINGS
50 TIMER OFF                :REM SHUT OFF TIMER UNTIL WANTED
60 GOSUB 390                :REM GO WAIT FOR A BUTTON OR TIMEOUT
70 ON TIMER(1) GOSUB 470    :REM HERE'S WHERE TO GO IF TIMEOUT
80 TIMER ON                 :REM START THE TIMER
90 LG=G:X=0                 :REM SAVE LAST BIT AND INIT COUNT
100 GOSUB 390               :REM GO WAIT AGAIN
110 IF T=1 THEN GOTO 30     :REM IF TIMEOUT THEN START OVER

120 IF (LG<>G) AND X>1 THEN GOTO 160 :REM START BIT RECOGNIZED
130 IF (LG=G) THEN X=X+1 ELSE X=0    :REM IF BIT THE SAME INCR.
140 LG=G                    :REM COUNT, SAVE THE BIT
150 GOTO 100                :REM GET ANOTHER
160 D=G                     :REM LAST BIT IDENTIFIES DIRECTION
170 FOR X=1 TO 16           :REM NOW FOR THE DATA BITS
180 GOSUB 390               :REM GET ONE
190 IF T=1 THEN GOTO 30     :REM TIMEOUT
200 IF (G=1) THEN S$=S$+"1" ELSE S$=S$+"0" :REM SAVE THE BIT
210 NEXT X                  :REM DO ALL BITS
220 GOSUB 390               :REM LOOK FOR THE END CODE
230 IF T=1 THEN GOTO 30     :REM TIMEOUT
240 IF (D=G) THEN GOTO 440:REM IF SAME THEN BAD END CODE
250 GOSUB 390               :REM NEXT BIT
260 IF T=1 THEN GOTO 30
270 IF (D<>G) THEN GOTO 440
280 GOSUB 390
290 IF T=1 THEN GOTO 30
300 IF (D<>G) THEN GOTO 440
310 GOSUB 390
320 IF T=1 THEN GOTO 30
330 IF (D<>G) THEN GOTO 440

340 PRINT
350 IF (D=0) THEN GOTO 370:REM NO SWAP NECESSARY IF DIRECTION OK
360 FOR Z=LEN(S$) TO 1 STEP -1: T$=T$+MID$(S$,Z,1):NEXT Z:S$=T$
370 PRINT S$                :REM PRINT THE DETECTED DATA
380 GOTO 30                 :REM LOOK FOR MORE
390 WHILE STRIG(1)=0        :REM DURING NO CLOCK MARK
400 IF T=1 THEN RETURN      :REM RETURN IF TIMEOUT
410 WEND
420 G=ABS(STRIG(5))         :REM NOW READ DATA
430 IF (STRIG(1)=-1) THEN GOTO 430 ELSE RETURN
                            :REM WAIT FOR NO CLOCK
440 REM BAD EXIT            :REM IF END CODE DOES NOT MATCH
450 PRINT".":               :REM WE MUST HAVE BAD DATA, INDICATE IT
460 GOTO 30                 :REM TRY AGAIN
470 REM TIMER OVERFLOW      :REM THIS IS THE TIMEOUT ROUTINE
480 TIMER OFF               :REM STOP TIMING
490 T=1                     :REM TIMER FLAG = TIMEOUT
500 PRINT "-";              :REM INDICATE IT
510 RETURN
```

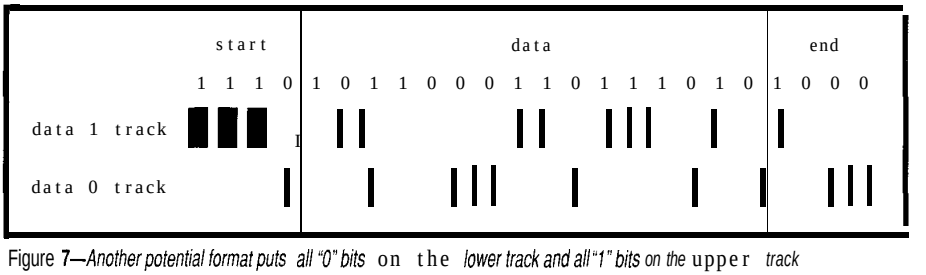


Figure 7—*Another potential format puts all "0" bits on the lower track and all "1" bits on the upper track*

accuracy. Although twice as many bits are packed into this format (as opposed to the clocked format), the throughput is the same, but now each bit is verified. Similarly, I've posted code similar to that in Listing lb to poll the PC's joystick port and display the received data bit sequence. If the start code, complemented data, and end code are received as expected, then a beep declares an accepted swipe.

On the most basic level, this optical reader could be used to recognize 1 of 65,535 different cards. Proper recognition might energize a solenoid-powered door lock or perform some other task designated by the card's code. The resolution of these sensors seems to be a 2-mm minimum space or mark. This can be improved slightly using a slotted mask at the focal point that does not allow adjacent marks from interfering with the total reflection.

Next month, I'll investigate trading cost for higher resolutions as well as adding some "smarts" to the Swipe reader.

Jeff Bachiochi (pronounced "BAH-key-AH-key") is an electrical engineer on the Computer Applications Journal's engineering staff. His background includes product design and manufacturing.

SOURCE

Digi-Key Corp.
701 Brooks Ave. South
Thief River Falls, MN 56701-0677
(800) 344-4539
Fax: (218) 681-3380

SOFTWARE

Software for this article is available from the Circuit Cellar BBS and on Software On Disk for this issue. Please see the end of "ConnecTime" in this issue for downloading and ordering information.

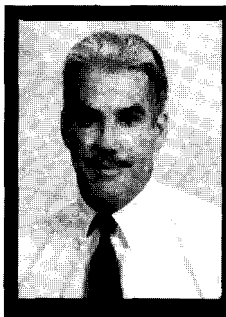
I R S

416 Very Useful
417 Moderately Useful
418 Not Useful

In Bed With PCS

SILICON UPDATE

Tom Cantrell



What's the best way to keep down both hardware and software costs when developing a product based on an embedded controller? For many, it's to use an IBM PC–compatible board. Find out what's on the market.

Ah the IBM PC—can't live with it, can't live without it. Yes, I admit to a distinct love/hate relationship with the PC.

On the one hand, I cut my micro teeth on the '86 family and, having designed boards and written a lot of assembly code, became intimate with its architecture-quirks, warts, and all.

But it's those same quirks and warts that can get to you. Starting with the in-your-face CISCiness of the CPU, strange and singular oddities percolate up through the PC system design and surface in the OS and application software.

Nevertheless, the PC's main virtue-its low price-makes up for myriad technical sins. Indeed, it's the incredible value of PCs that is driving the "downsizing" in the computing market. The same forces are at work in the embedded arena.

SO MANY PCs, SO LITTLE TIME

Deciding if an embedded PC is right for you depends on whether certain key application characteristics match the strengths of the PC, while avoiding its weaknesses. Of course, even with insane discounting, some low-end applications don't need the expandability and can't afford the overhead associated with a PC. However, with ever falling prices, it's more and more likely an embedded PC is in your future.

For instance, any situation that calls for a disk or CRT is likely well served by an embedded PC. Why reinvent the wheel when you can toddle over to your local PC-To-Go emporium and pick up a VGA monitor or a hard disk for a song? In fact, the advantage extends to nearly any mass storage (floppy, CD-ROM, tape] and display (CRT, LCD, plasma) technology.

Potential trouble spots are applications that demand speedy real-time response or a large linear address space. Actually, the performance limits are mainly a function of software like BIOS, DOS, and Windows, and can be overcome to some degree by writing or buying different software. However, I feel that compatibility with PC development tools is a major advantage, and when it is lost, the embedded PC approach starts to make less sense.

As this by-no-means exhaustive buyers guide indicates, there is a

Photo I--The *Industrial Computer Source 6200-UMR rack-mount packaging system.*

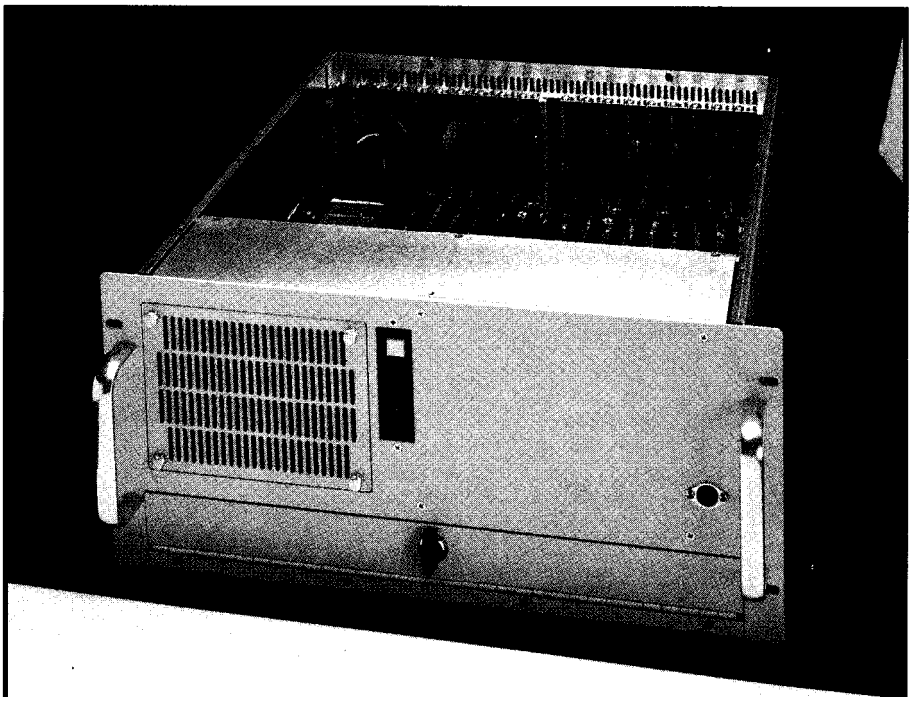


Photo **2a**—*The IPC-616* from *Advantech includes a 16-slot motherboard and a 250-W power supply in ifs 19″ rack-mount chassis.*

bewildering array of products that fall under the heading of "embedded PC." To help make sense of it all, I classify them into the following groups:

*PC-In-A-Box: Factory floor enclosures for standard desktop PCS.

*Passive Backplane: PC bus (typically ISA) plug-in CPU boards.

- Alt-Bus: Passive backplane or mezzanine non-PC bus boards.
- Almost-PC: PC "developable," but not PC compatible.

Read on to compare and contrast the strengths and weaknesses of each flavor.

PC-IN-A-BOX

Once, visiting the factory floor of an industrial concern, I watched as a maintenance technician approached an imposing and rather expensive looking control system packaged in a jukebox-sized rack. As the technician prepared to open the access panel, I expected to see some fancy electronics. Imagine my surprise when I spied, resting on the floor of the cavernous-and otherwise empty-bay, a noname PC clone.

You can give your PC a big-iron makeover without needing a machine shop. For instance, Industrial Computer Source offers the 6200-UMR rack-mount packaging system (Photo 1) that accommodates "small footprint" PCs, keyboards, and desktop monitors.

The PC-In-A-Box approach has the advantage of minimizing the cost of the PC portion. After all, no alternative implementation of the PC can match the pricing of a PC itself.

On the other hand, appearances to the contrary, a rugged package does not a rugged PC make. Make sure your setup can meet the hermetic, temperature, and vibration specs demanded by your application. Also, a desktop PC doesn't lend itself to easy maintenance as anyone who has had to do a motherboard swap knows.

PASSIVE BACKPLANE PCs

For a more robust, but still completely hardware and software compatible alternative, consider a passive backplane PC. As the name implies, this scheme adopts the traditional card cage approach in which all the PC motherboard logic is scooped onto a PC bus (whether it's AT/ISA or even EISA or MCA) plug-in board. Compared to a desktop PC, a motherboard swap for purposes of maintenance or upgrade becomes a 60-second, rather than 60-minute, proposition.

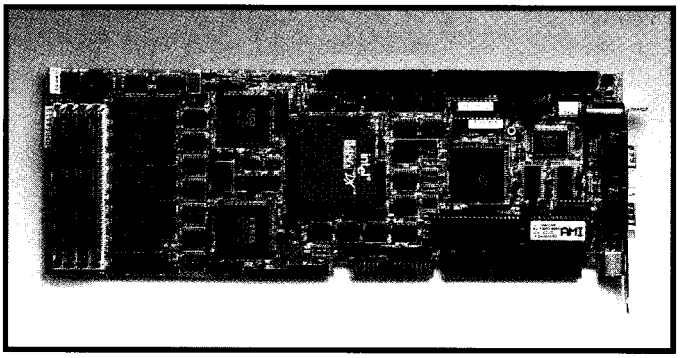


Photo **2b**—*The Advantech PCA-6146 processor board includes a '486DX, up to 16M of DRAM, 256K of cache, IDE and floppy interfaces, serial and printer ports, battery-backed real-time clock, and watchdog timer.*

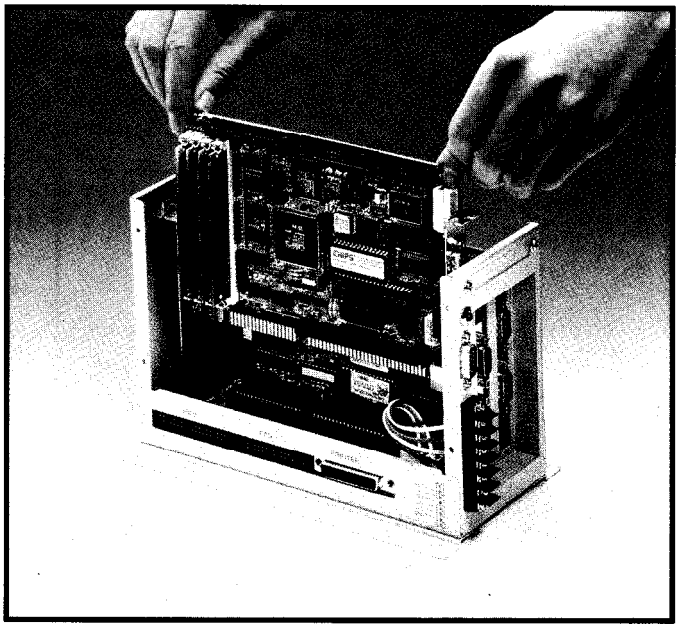


Photo **2c**— *When space is an issue, the Advantech MBPC-640 with ifs three half-size slots is sufficient for many applications.*

One of the advantages of the passive backplane approach is a wide range of size and expandability options. For example, consider the range of offerings from Advantech.

The IPC-616 packs a 16-slot motherboard and a hefty 250-W power supply into a 19" rack-mount chassis (Photo 2a). A correspondingly beefy CPU is the PCA-6146 (Photo 2b), which matches the specs of top end desktop computers—'486DX, up to 16M of DRAM, 256K of cache, IDE and floppy interface, two serial ports, one printer port, battery-backed RTC, and even a 1.5/15/150-second selectable watchdog timer.

At the other extreme, consider the MBPC-640 (Photo 2c) which, thanks to the ever shrinking VLSI, can actually handle significant applications with a measly three half-size slots. Use one slot for the PCA-6134-33 386SX CPU card and you've got two left for your applications' unique I/O needs.

ALT-BUS PCs

This refers to systems that offer complete PC software and functional compatibility, but are based on a non-PC bus. Of course, most of the differentiation depends on the characteristics of the particular bus used.

You can choose a "standard" bus such as VME, STD 32, or the new PC/104, all of which offer multivendor mix and match capability. This is especially useful if you must interface to existing boards or systems that use a particular bus.

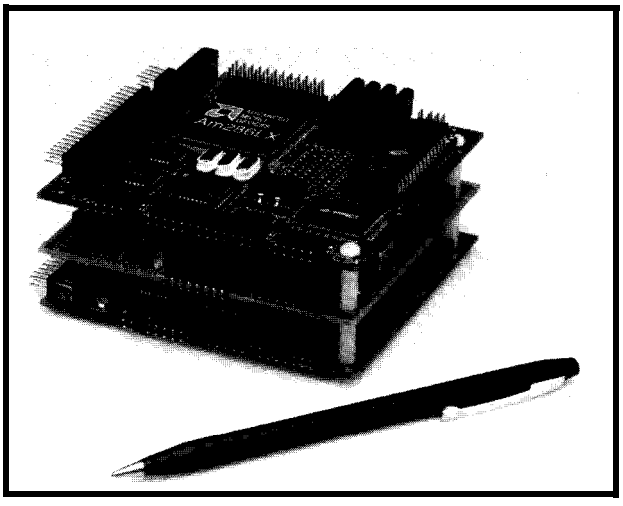


Photo **3b**—*A typical PC/104 stack from Ampro consists of a '286 CPU board, an Ethernet interface, and a I/GA controller.*

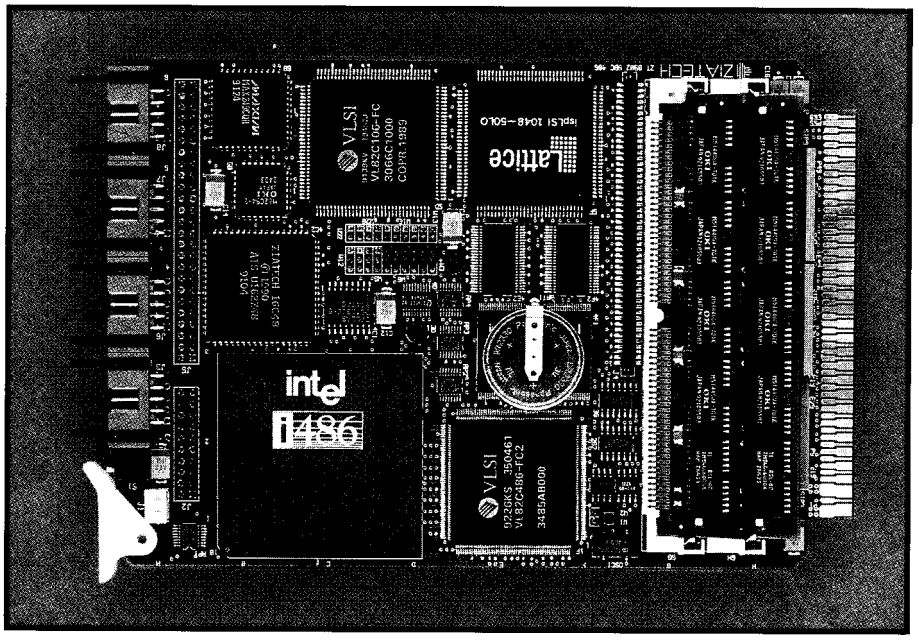


Photo **3a**—*The Ziatech 8902 STD 32 embedded PC includes support for a plug-on super VGA adapfer.*

The STD 32 bus is a clever 32-bit upgrade of the old standby 8-bit STD bus. Unlike the ISA bus, which differentiates 8-bit and 16-bit slots with an extra connector, STD 32 interleaves the new 32-bit signals with the old 8-bit signals. This allows flexible setup, including a 32-bit CPU in an 8-bit bus, an 8-bit I/O board on a 32-bit bus, and, of course, a full 32-bit configuration. A key benefit relative to ISA, with its 8/16-bit connector and full/half slot dilemmas, is that all STD/STD 32 boards are the same size and thus can be fully supported on all sides. Boards flapping in the breeze are particularly a no-no if vibration is an environmental concern. Photo 3a is an example of a STD 32 embedded PC, in this case the Ziatech 8902 with plug-on super VGA adapter.

The latest standard contender is the PC/104 spec, which is being proposed as an extension to the IEEE P996 (draft) ISA specification. Unlike all the other buses, PC/ 104 is a mezzanine-type (stackable) bus featuring, like ISA, either one (P1/8-bit) or two (P1&P2/16-bit) connectors. Photo 3b shows a typical PC/104 "stack" from Ampro consisting of a '286 CPU board, an Ethernet interface, and VGA controller.

Configuring a system does take a little thought to meet the constraints of the mezzanine scheme. For example, 8-bit boards need to be on top (since they don't pass the 16-bit signals on) and a stack can only handle a single "high profile" board (for example, a relay board) on top. Also, the appropriate mix of "stackthrough" and "nonstackthrough" connectors is called for. Debugging and maintenance is complicated by lack of access to the innermost cards.

The PC/104 approach has the advantage of small-size and, with the appropriate spacers, good rigidity. Since it is electrically quite similar to the ISA bus, the many members of the consortium are hard at work "porting" existing ISA bus designs and chipsets to PC/104.

As an alternative to a standard like VME, STD 32, or PC/104, you can go with a particular company's "proprietary" bus should it offer functional advantages such as especially small size or unique packaging.

For example, the E.S.P. (Extremely Small Package) line from DOVatron (formerly Dover Electronics Manufacturing) combines small form factor (at

1.7" x 5.2", even smaller than PC/104) with downsized plug-in backplane packaging (Photo 4a).

Meanwhile, Micro-Link takes another tack by mapping the ISA bus signals onto the 3U (single height) Eurocard format with its robust DIN connector and four-sided mounting stability (Photo 4b). This combines the best of both worlds by exploiting heavy-duty and standardized Eurocard packaging while allowing the use of low-cost ISA boards and chipsets.

Finally, the Radisys EMC packages the PC in a true industrial strength package with extended temperature, shock, and vibration specifications (Photo 4c). In fact, the EMC even complies with military specifications for rugged instruments.

The only caveat with the proprietary route is that everything is sole-sourced. However, be reassured by the fact that suppliers realize they have to offer a complete selection of add-ons at a competitive price.

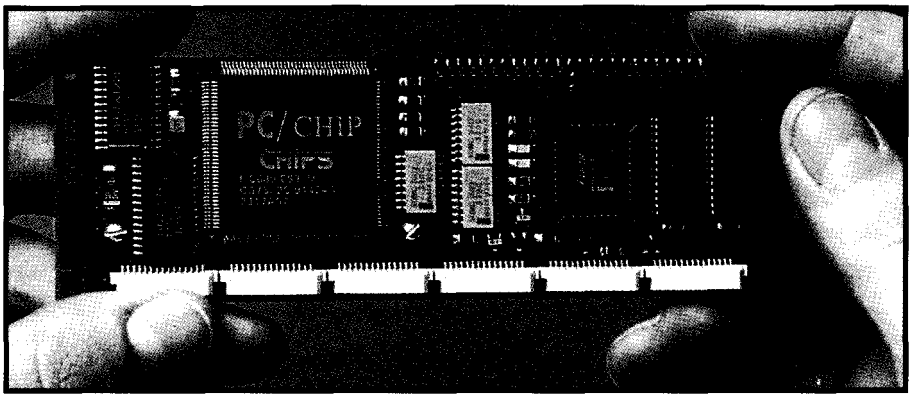


Photo **4a**—*The Extremely* Small *Package (E.S.P.) line from DOVatron combines small form factor with conventional plug-in backplane packaging.*

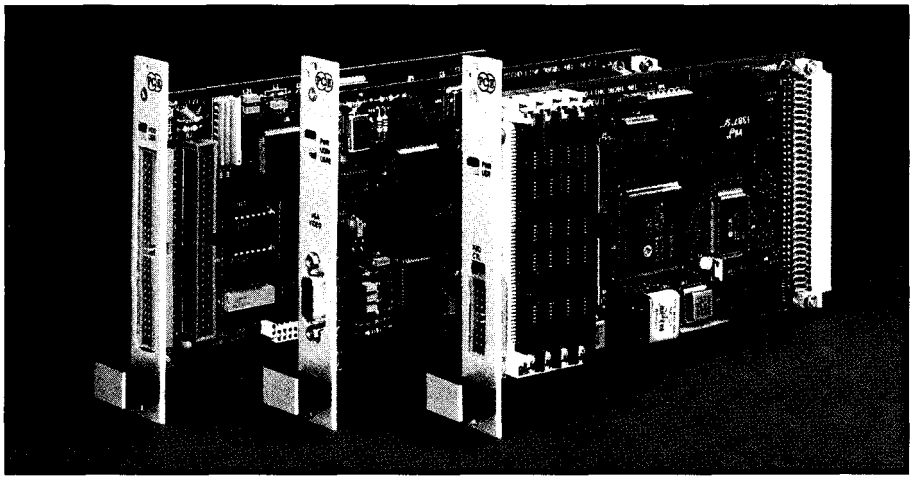


Photo **4b**—*Microlink maps the ISA bus signals onto the 3U Eurocard format.*

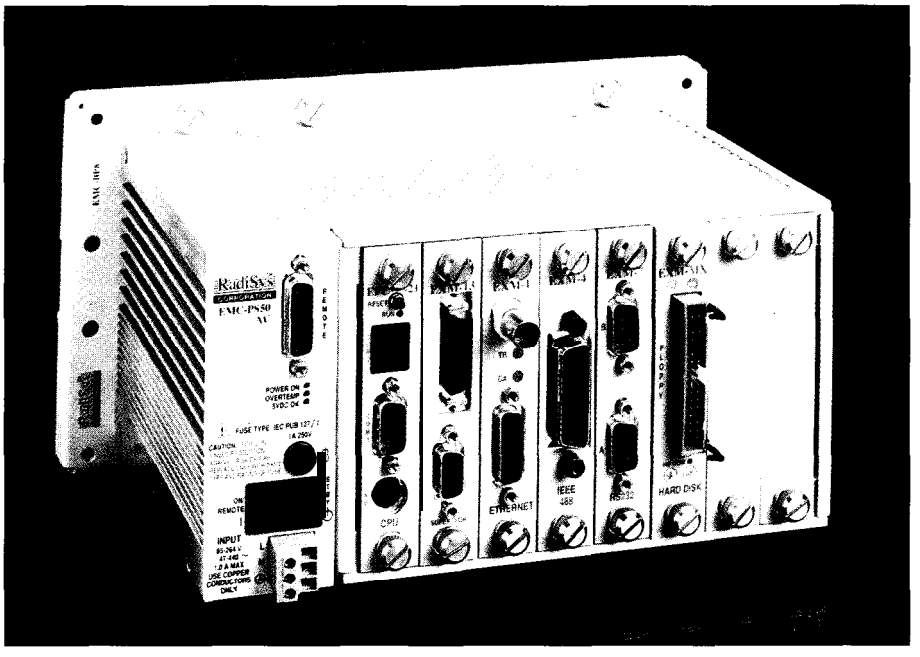


Photo **4c**—*The Radisys EMC family packages the PC in a true industrial strength unit with extended temperature, shock, and vibration specifications.*

ALMOST-PCs

Besides price and hardware expandability, there is a class of embedded designs that exploits the final advantage of PCs-great development tools.

These PC pretenders trade off strict compatibility in favor of low-cost, small size, low power, and industrial-type I/O. Unlike the other approaches, you shouldn't even try to configure a true PC with this technology which, needless to say, fails the "Flight Simulator test."

Rather, boards like the Micromint RTC-V25 and R.L.C. Enterprises Mini-Cl86 are only designed with enough compatibility to allow the use of popular PC-based tools such as Borland C. The embedded and desktop PCs are linked with a serial port and packages-such as those from Paradigm, Datalight, and others-that enable source-level debugging of code executing on the target.

The RTC-V25 (Photo 5a) combines the NEC 8088-like V25 CPU with 32 parallel I/O lines, an S-channel ADC (8- or lo-bit), battery-backed clock/calendar, 128 bytes of EEPROM, two serial ports, and a mix of up to 384K RAM or ROM. Roughly the same size as a PC/104 card, the RTC-V25 also has a stackable, though proprietary, bus for I/O expansion. As for add-on boards, there's nary a VGA, IDE, or game port to be found. Instead, there are control-oriented expansion boards such as TTL, buffered, and optoisolated I/O;12-bit ADC; LCD; infrared; and so forth.

The Mini-Cl86 (Photo 5b) flaunts its difference from the pack with its nickname as the "No Bus-No Fuss" computer. It combines a '186 with three 16-bit timers, two serial ports, watchdog timer/power fail detect, program-accessible DIP switch and LEDs, and up to 512K each of EPROM and SRAM. Despite the "No Bus" moniker, it also includes two iSBX connectors for modules adhering to that Intel-defined I/O add-on standard.

PICK A PECK OF PCs

Understanding the merits of each category of PC makes choosing the right alternative a little easier.

The particular need for a disk and/or CRT is a vote in favor of the "PC-In-A-Box" strategy because you can take advantage of the competition in the desktop market and get a really low price. Since the disk and/or CRT dictate a fairly benign environment, the fact a desktop PC isn't that rugged is moot. Frankly, this approach is also suitable in cases where the customer will be happier paying the bill if they perceive they are getting a "hunk of iron" rather than a plain old PC like the one they gave their kid for Christmas. Of course, nothing's more PC compatible than a PC itself.

If you especially need or want to use PC add-on boards and are willing to pay for easier upgrades and maintenance, consider the "Passive Backplane" approach. Particularly if a CRT and/or disk isn't part of the picture, you can configure a hardened system with beyond desktop temperature and vibration tolerance. Naturally, this approach is also fully PC compatible since it involves little more than making the CPU board a "plugger" rather than a "pluggee."

If you need full PC software compatibility in a smaller and/or more rugged form-factor, consider one of the many "Alt-Bus" alternatives. Choose a standard bus-such as STD 32 or PC/104-if you need a broad variety of I/O add-ons from multiple suppliers. Or, go with a particular company's proprietary bus if it's an ideal match for your application.

If you really just want to use PC development tools, and find the above

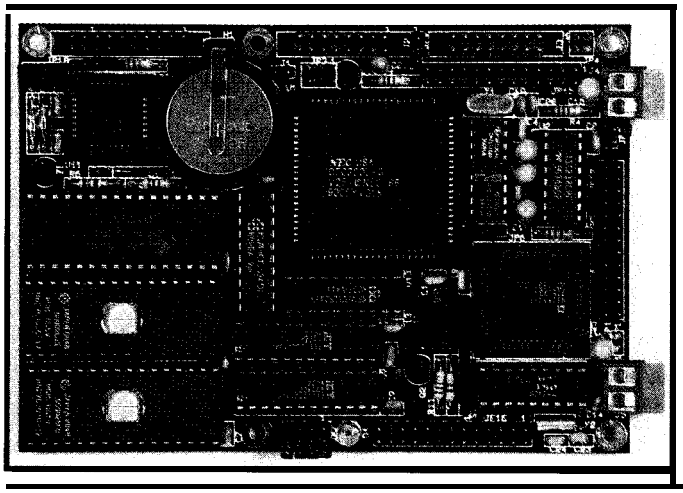


Photo **5a**—*The Micromint RTC-V25 combines the NEC 8088-like V25 CPU with 32 parallel I/O lines, an d-channel ADC, battery-backed real-time clock/calendar, 128 bytes of EEPROM, two serial ports, and a mix of up to 384K RAM or ROM.*

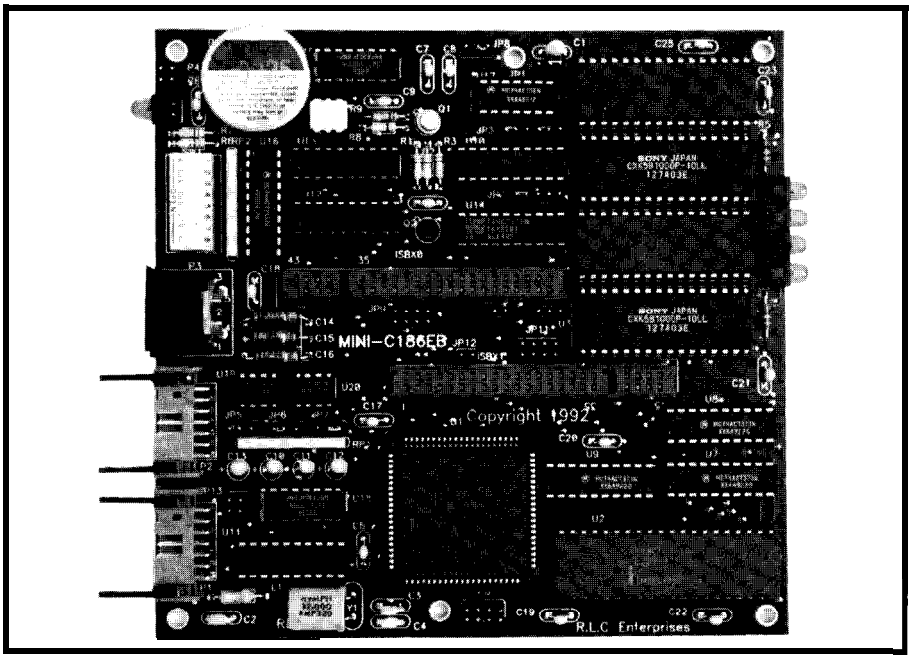


Photo **5b**—*The* R.L.C. Enterprises Mini-Cl86 promotes *"no* bus, no fuss" by packing everything onto a *single* board.

approaches offer a lot of stuff you don't need (VGA, floppy, game port, etc.) and not the stuff you want (buffered or optoisolated I/O, ADC, etc.) consider an "Almost PC." If you're lucky, you'll find a board that's just what you need-no more, no less-with the lowest cost, size, and power consumption. ❑

Tom Cantrell has been an engineer in Silicon Valley for more than ten years working on chip, board, and systems design and marketing. He can be reached at (510) 657-0264 or by fax at (510) 657-5441.

IRS

419 Very Useful
420 Moderately Useful
421 Not Useful

Embedded PC Roundup

Advantech
750 East Arques Ave.
Sunnyvale, CA 94086
(408) 245-6678
Fax: (408) 245-8268
Passive-backplane CPUs, I/O boards, and packaging

Amdex
76 Treble Cove Rd.
N. Billerica, MA 01862
(508) 663-2070
Fax: (508) 663-5094
Passive backplane CPUs, I/O boards, and packaging

Ampro Computers, Inc.
990 Almanor Ave.
Sunnyvale, CA 94086
(408) 522-2100
Fax: (408) 720-1305
PC/104 CPU and I/O boards

Analogic Corporation
360 Audobon Rd.
Wakefield, MA 0 1880
(508) 977-3000
Fax: (617) 245-1274
Passive backplane I/O boards and packaging

Annabooks
15010 Ave. of Science, #101
San Diego, CA 92128
(619) 673-0870
Fax: (619) 673-1432
Embedded PC documentation and training

APPRO International
3687 Enochs St.
Santa Clara, CA 9505 1
(408) 732-6091
Fax: (408) 732-6095
Passive backplane CPUs, I/O boards, and packaging

Automated Control Concepts
3535 Route 66
Neptune, NJ 07753
(908) 922-6611
Fax: (908) 922-9611
Passive backplane system with steel NEMA 4/12 packaging

Azimuth Technologies
6 Landmark Sq., 4th floor
Stamford, CT 06901
(203) 359-5706
Passive backplane 486 system and rack-mount packaging

Computer Boards, Inc.
44 Wood Ave.
Mansfield, MA 02048
(508) 261-1123
Fax: (508) 261-1094
Analog and digital I/O boards, Opto-22 interface and racks

Computer Dynamics
107 S. Main St.
Greer, SC 29650
(803) 877-8700
STD bus and stand-alone PC compatible SBCs

CyberResearch, Inc.
P.O. Box 9565
New Haven, CT 06535-0565
(203) 483-8815
Fax: (203) 483-9024
Passive-backplane CPUs, I/O boards, and packaging

Daisy Data, Inc.
333 South Enola Dr.
Enola, PA 17025
(717) 732-8800
Fax: (717) 732-8806
NEMA factory floor packaging for PCs and workstations

Datalight
307 N. Olympic, #201
Arlington, WA 98223
(206) 435-8086
Fax: (206) 435-0253
Software adapts PC programming tools to work with embedded PCs

Dover Electronics Mfg.
1198 Boston Ave.
Longmont, CO 80501
(303) 772-5933
Tiny 1.7" x 5.2" CPU and I/O modules

ERIM
(602) 962-5559
Fax: (602) 962-5750
PC/104 CPU board based on C&T PC/CHIP

General Software
P.O. Box 2571
Redmond, WA 98073
(206) 391-4285
Fax: (206) 557-0736
DOS and BIOS tailored for use with embedded PCs

HM Systems, Inc.
2192 Dupont Dr., #214
Irvine, CA 927 15
(714) 955-2043
Fax: (714) 955-1849
'386 and '486 passive backplane CPU boards

I-Bus
9596 Chesapeake Dr.
San Diego, CA 92123
(800) 382-4229
Fax: (619) 974-6494
Passive backplane CPUs and board packaging

Industrial Computer Source
P.O. Box 23058
San Diego, CA 92 193
(619) 271-9340
Fax: (619) 271-9666
Passive-backplane CPUs, I/O boards, and packaging

Innovative Integration
4086 Little Hollow Pl.
Moorpark, CA 93021
(805) 529-7570
Fax: (805) 529-7932
TMS320 DSP-based ISA bus coprocessor

Intecolor
2 150 Boggs Rd.
Duluth, GA 30136
(404) 623-9145
Fax: (404) 623-9163
Passive backplane CPUs, I/O boards, and packaging

Integrated Systems, Inc.
3260 Jay St.
Santa Clara, CA 95054-3309
(408) 980-1500
Fax: (408) 980-0400
Real-time OS for embedded PCs and other systems

Interlogic Industries
85 Marcus Dr.
Melville, NY 11747
(516) 420-8111
Fax: (516) 420-8007
Passive backplane (ISA & EISA) CPUs, backplanes, packaging

JF Microsystems
3641 Frontier Rd.
Pasco, WA 99301
(800) 532-2737
STD bus CPU and I/O boards

Logical Design Group, Inc.
6301 Chapel Hill Rd.
Raleigh, NC 27607
(919) 851-1101
Fax: (919) 851-2844
PC-compatible CPU boards with VME bus interface

MCSI
2598G Fortune Way
Vista, CA 92083
(800) 347-MCSI
ISA- and EISA-based SBCs

Megatel Computer Corp.
125 Wendell Ave.
Weston, Ont. M9N 3K9 Canada
(416) 245-2953
Fax: (416) 245-6505
PC/ 104 and Eurocard CPU and I/O boards

Mesa Electronics
1329D 61st St.
Emeryville, CA 94608
(510) 547-0837
CPU board combined with CGA backlit LCD

Micromint, Inc.
4 Park St.
Vernon, CT 06066
(203) 871-6170
Fax: (203) 872-2204
3.5" x 5" V25-based SBC.

Micro-Aide
685 Arrow Grand Cir.
Covina, CA 9 1722
(818) 915-5502
STD bus V40 and 386SX-based SBCs

Micro Alliance, Inc.
449 Santa Fe Dr., Ste. 800
Encinitas, CA 92024
(619) 942-9800
Passive backplane CPUs, I/O boards, and packaging

Micro Link
401 Pennsylvania Pkwy., Ste. 205
Indianapolis, IN 46280
(800) 428-6155
Eurocard, VME, and STD format PC and I/O boards

Micro/Sys
3447 Ocean View Blvd.
Glendale, CA 91208
(818) 244-4600
STD bus '386 and '486 DOS-compatible SBCs

Multi-Micro Systems
62 Bonaventura Dr.
San Jose, CA 95 134
(408) 456-0333
Fax: (408) 456-0366
Passive backplane CPUs, I/O boards, and packaging

Octagon Systems
6510 W. 91st Ave.
Westminster, CO 80030
(303) 430-1500
Small form-factor (4.5" x 5") PC-compatible SBCs

Paradigm Systems
3301 Country Club Rd., #2214
Endwell, NY 13760
(607) 748-5966
Fax: (607) 748-5968
Software adapts PC programming tools to work with embedded PCs

Pro-Log Corporation
2555 Garden Rd.
Monterey, CA 93940
(800) 5389570
STD bus CPU and I/O boards

PC/104 Consortium
990 Almanor Ave.
Sunnyvale, CA 94086
(408) 245-9348
Fax: (408) 720-1322
Licensing organization—contact for complete list of PC/104 suppliers

Radisys Corp.
15025 SW Koll Pkwy.
Beaverton, OR 97006
(503) 646-1800
Small form-factor CPU and I/0 modules with rugged packaging

Recortec, Inc.
1290 Lawrence Station Rd.
Sunnyvale, CA 94089
(408) 734-1290
Fax: (408) 734-2140
Rackmount and factory floor enclosures

R.L.C. Enterprises
4800 Templeton Rd.
Atascadero, CA 93422
(805) 466-9717
"No-bus"/"No-DOS" 80186-based SBC

Saelig Company
(716) 425-3753
Fax: (716) 4253835
3.4" x 2.6" PC/AT compatible

Spectrum Controls
P.O. Box 5533
Bellvue, WA 98006
(206) 746-948 1
Fax: (206) 641-9473
CPU board packaged with LCD and keypad

STD 32 Special Interest Group
11766 Wilshire Blvd., Ste. 370
Los Angeles, CA 90025
(800) 733-2111
Fax: (800) 733-3959
Licensing organization for STD 32 bus designs

Symmetric Research
15 Central Way, #9
Kirkland, WA 98033
(206) 828-6560
Fax: (206) 827-3721
DSP32C-based PC bus coprocessor

Systek
415 N Quay St., Ste. 6
Kennewick, WA 99336
(509) 7351200
V25- and V50-based SBCs

Systronix, Inc.
555 South 300 East
Salt Lake City, UT 84111
(801) 534-1017
Fax: (801) 534-1019
SBC with 8088-type CPU, temp sensor, and 2 x 24 LCD

Teknor Microsystems
P.O. Box 455
Sainte Therese, Quebec
Canada, J7E 4J8
(514) 437-5682
Fax: (514) 437-8053
Passive backplane CPU and board packaging

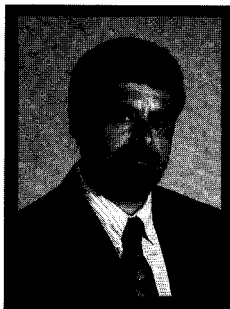
Texas Micro
P.O. Box 42963
Houston, TX 77242-9910
(800) 627-8700
Fax: (713) 541-8226
Passive backplane packaging and boards

Versalogic Corp.
3888 Stewart Rd.
Eugene, OR 97402
(800) 824-3163
STD-32 CPU and I/O boards

WinSystems, Inc.
715 Stadium Dr. #112
Arlingon, TX 760 11
(8 17) 274-7553
PC/104 CPU and I/O boards

Ziatech Corp.
3433 Roberto Ct.
San Luis Obispo, CA 93401
(805) 541-0488
Fax: (805) 541-5088
STD-32 bus and ISA CPU and I/O boards

Support For Your Batteries



Just charge it. The question is, which type. John concludes his series on battery management by discussing several charging techniques using Benchmarq's line of charger chips.

EMBEDDED TECHNIQUES

John Dybowski

When thinking about this month's article, I considered a cursory survey of all the available battery support peripherals on the market. I thought I could highlight and compare some of their more prominent features. Instead, I decided to narrow my focus and cover just a few parts so I could provide more in-depth details.

I really like what Benchmarq is doing, so I'll concentrate on their offerings. Having started with the bq2003 fast-charge IC last month, I'll continue now and elaborate more fully on its capabilities and show you how to use it in several different charging configurations.

BQ2003 RECAP

The bq2003 is a complete fast battery charger circuit suitable for NiCd, NiMH, or lead acid batteries. It can operate in a stand-alone fashion or can be embedded into a battery operated system as an integrated function block. Charging current can be generated by use of a built-in, high-efficiency, switched-mode current regulator, or an external linear current source can be gated through the device to provide charging current. LED drivers for displaying battery and charge status are built into the device. A single LED shows events such as charge pending, discharge, fast charge in progress, charge complete, and charge aborted. They pack all of these different status indications into a single LED by driving it using a varying duty cycle sequence of on/off pulses. This kind of approach (when taken to extremes) can degenerate into an incomprehensible rash of gibberish that ultimately conveys nothing. In this usage, however, it is not a problem at all to decipher the status being conveyed since only a few different (and rather distinct) patterns are issued. Take this courteous treatment of the end user as an object lesson in judicious restraint and good design practice. Temperature status is also shown (using a separate LED) that indicates an out-of-range temperature when it is illuminated. In an attempt to clarify the bq2003's operation, let me begin with an overview of the IC's pin functions presented in Figure 1.

Charge action is controlled by inputs from the CCMD (charge command), DCMD (discharge command), and DVEN (negative delta voltage enable) input pins and the TM1 and TM2 (failsafe timer/initial hold-off interval/top-off enable) programming pins. Charge initiation is qualified by two factors. First, the battery temperature must be between the low temperature fault and high temperature fault levels. Second, the voltage of the cell must be between the end-of-discharge voltage and the maximum cell voltage. If a discharge-before-charge cycle is selected, it is performed prior to initiation of fast charging. Once fast charging begins, delta temperature/delta time and/or negative delta voltage are monitored to determine when a full charge has been reached.

Temperature cutoff, maximum voltage, and maximum time are tested in order to stop the fast charge if, for any reason, the primary cutoff mechanisms should fail. This redundant testing method is used for fail-safe operation. Of course, under normal conditions the primary fast charge cutoff mechanism should work just fine, but the experienced engineer understands the need for backup schemes. This understanding often overcomes the uninitiated right about the time they move their designs from the sterile confines of the workbench into the cold, cruel world. Frequently, this experience is amplified when the single prototype suddenly spawns a bunch of production units that find their way into less-than-friendly

environments. In any event, realize that the primary charge determination signals operate at relatively low level voltages. It would be a shame if a glitch or some other system anomaly caused costly batteries to dry up or outgas. These backup schemes exist for a reason.

With a general understanding of the bq2003's pin functions and charging strategy, refer to Figure 2 for more details of the bq2003's actual charge sequence.

LINEAR CONSTANT CURRENT

As you know, the bq2003 can be configured to generate charging current using its built-in, buck-type, switch-mode controller. Although much more efficient than a linear current source, it is admittedly more expensive to put together. For relatively low charging currents, a linear constant-current source usually works fine. Nonetheless, it's wise to consider the current and power requirements along with the prevailing thermal issues before erring on the side of simplicity. When the current requirements fall below a certain threshold, the choice becomes much more clear cut-and safe.

Utilizing the flexibility of the bq2003, you can still enjoy the benefits of the superior delta temperature/delta time and negative delta voltage charge termination mechanisms while using a simple, and cheap, linear constant-current source. The bq2003's cost/feature ratio can easily justify using only a portion of its capability. Of course, the safety backup and charge disable functions along with the discharge-before-charge capability are still available even if you decide to go with this simpler configuration.

Referring to Figure 3, you can see by connecting SNS to ground, MOD gates an external current source for the duration of the charging sequence until a terminating event is detected. In this arrangement, taking MOD high turns Q2 on, which removes bias from Q3. This enables the constant-current source, which is based on the ubiquitous LM317 (U2). Using the formula I=1.25V/R16, the current can be set up to a maximum of 1.5 A. In this type of brute-force configuration, you must watch your power.

BAT	**Single-cell battery voltage input** A voltage level developed by a high-impedance resistor divider between the positive and negative battery terminals that sets the single-cell voltage for the battery being used.
MCV	**Maximum cell voltage threshold input** The voltage at this pin sets the maximum single cell voltage.
TS	**Temperature sense input** Connection to an external battery temperature monitoring (negative temperature coefficient [NTC]) thermistor.
TCO	**Temperature cutoff threshold input** The voltage at this pin sets the maximum allowable battery temperature.
CCMD, DCMD	**Charge initiation and discharge-before-charge control inputs** When both CCMD and DCMD are at VCC or when both are connected to VSS, charge initiation is automatically started on battery replacement or application of VCC. Charge is also initiated by a rising edge to VCC at CCMD if both CCMD and DCMD are connected to VSS, or by a falling edged on CCMD if both CCMD and DCMD are connected to VCC. Discharge-before-charge is initiated by a rising edge at DCMD if both DCMD and CCMD are connected to VSS, or by a negative-going edge on DCMD if both DCMD and CCMD are connected to VCC.
DVEN	**Negative delta voltage enable input** If this input is high, negative delta voltage charge termination is enabled.
DIS	**Discharge FET control output** An active-high push-pull output used to turn on an external transistor to discharge the battery through an external load before charging.
TEMP	**Temperature status output** An active-low push-pull output that indicates when the battery temperature is not within the acceptable range to initiate charging.
CHG	**Charging status output** A push-pull output used to indicate charging status.
TM1, TM2	**Timer mode inputs** These three-level inputs control the settings for the fast charge safety timer, initial termination monitoring hold-off interval, and select the "top-off" capability.
MOD	**Current switching control output** An active-high push-pull output that controls charging current to the battery.
SNS	**Charge current sense input** This input controls switching of MOD based on an external sense resistor. If SNS is connected to VSS (external current source mode), MOD switches high at the start of the charge cycle and low at the end of the cycle.

Figure 1—*The bq2003 handles a number of different charging schemes plus provides feedback to the user of current charge and temperature status.*

Charge can be initiated on battery replacement or by VCC going valid. In this particular arrangement, negative delta voltage detection is enabled (DVEN high), and discharge before charge is disabled (DCMD low]. The delta temperature/delta time threshold is set to 1.04°C per minute, and the high temperature cutoff is set to 50°C. The charging prequalification parameters configure the low temperature fault to 10°C with the high temperature fault set to 47°C. Not shown are the connections to TM1 and TM2, the safety time/hold off interval/top off selection pins, as well as the value for the trickle current resistor (R10). Select R10 for the trickle current rate that meets your particular application.

Note that the trickle resistor serves two purposes in the charging system. As you'd expect, it supplies a small trickle current that provides a charge-sustaining current once the fast

charge cycle completes. It also sources the current required to condition a deeply discharged battery prior to the application of a fast charge. The second usage of the trickle resistor is to provide a high-voltage supply that is used as a reference which allows the bq2003 to detect a battery insertion.

Incidently, if you set up the chip to use the top-off feature that delivers charging current at a reduced duty cycle (1/8 the fast charge rate), the trickle resistor can be a fairly large value since it will only have to put back the energy lost to self discharge. You may wish to keep this fact in mind when working with NiMH batteries because they are less tolerant of overcharge than NiCd types.

When selecting the main power supply, make sure to account for the voltage drop across the LM317 (U2), the blocking diode (D6), and the current selection resistor (R16). Adding these losses together, the total figure comes to about 4.5V. Add to this the number of cells times the maximum cell voltage to arrive at the minimum input voltage required to deliver current to the battery as it approaches full charge.

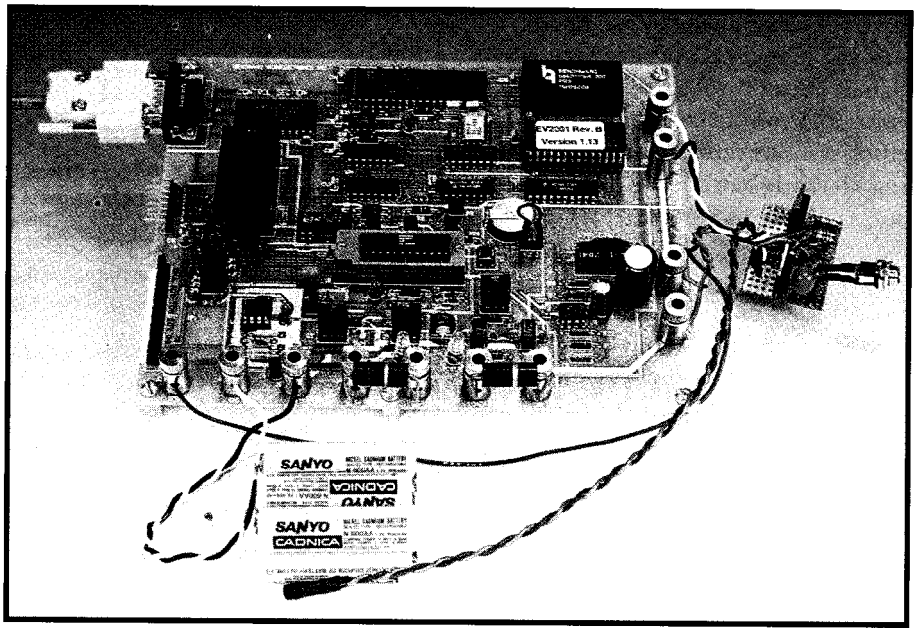


Photo 1—*The bq2001 charge management IC has so many features and options, a complete development system is available for it.*

SWITCHED CONSTANT CURRENT

The bq2003 can be configured as a switched mode current driver that is much more efficient than a typical linear current source. If the main power supply's current limitation or power limitation is a problem, then a switching current source may be the only way to go. Even if a hefty power supply is available, the product packaging might not tolerate the heat buildup associated with a linear approach.

Using external switching transistors, the bq2003 can be set up to operate with either a p-channel or an n-channel output power stage. For charging currents below 3 amps, a p-channel output stage is usually used since fewer support components are required. If the charge current is above 3 amps, using an n-channel FET usually turns out to be more economical even though additional parts are needed to establish the proper gate drive for the n-channel FET.

Figure 4 shows a complete charging system based on a p-channel switching element. This configuration can handle from 4 to 12 NiCd or NiMH cells at currents up to 3 amps.

Here, MOD drives a small-signal DMOS FET (Q3) that turns on when MOD is high. This FET, in turn, drives the p-channel power FET (Q1) into enhancement. Current through the inductor ramps up and the resulting voltage developed across the sense resistor (R26) is delivered to SNS via an R/C network composed of R4 and C8. When SNS reaches 0.250 volts, MOD goes low and the p-channel FET (Q1) turns off. At this time, a flux reversal occurs in the inductor causing the catch diode (D10) to conduct. Charge

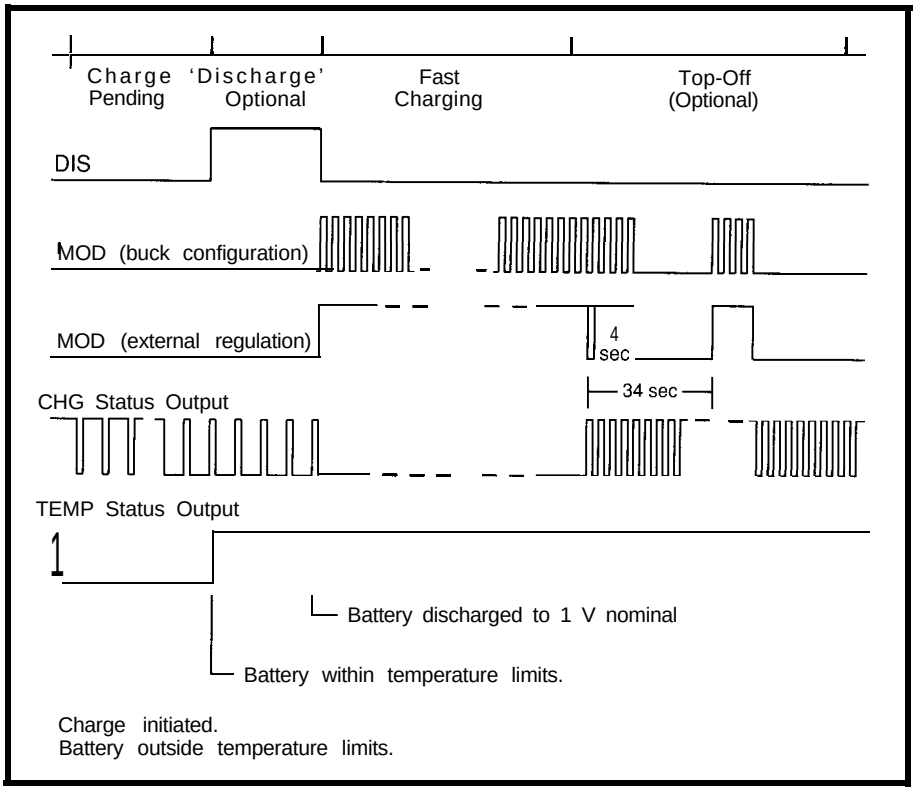


Figure 2—*The bq2003 will automatically handle all phases of a battery charge cycle and provides feedback to the user by flashing an LED at varying rates.*

current is delivered to the battery until the inductor current ramps down and the voltage at SNS reaches 0.220 volts. The cycle now repeats with MOD going high. From this description, you can see how the linear configuration described above switches charging current continuously by simply pulling SNS to ground.

For currents in the range of 3-9 amps, an n-channel power stage is usually employed as shown in Figure 5. Although requiring additional support components, the n-channel topology offers a price/performance advantage at these higher current levels. The n-channel's gate must be driven positive with respect to the drain in this configuration in order to provide full enhancement of the power FET (Ql). This is accomplished with the charge pump made up of C10 and **Cl 1.** When the catch diode (D10) is conducting, Cl 1 is charged. When the n-channel power FET is conducting, C **11** charges C 10 providing adequate voltage to fully enhance the power driver (Ql) via Q5. When Q2 conducts, gate charge is depleted thereby turning Q1 off. In all other respects, this circuit resembles the one based on the p-channel driver.

THE WELL-CONDITIONED BATTERY

Maximum battery capacity and cycle life are both dependent-to a great degree-on properly limiting heating during charging. This limiting can be achieved by using a fast reliable method of minimizing overcharge at fast charge rates. In the case of NiCd and NiMH batteries, the method used to apply the constant-current charge can also affect the overall charging efficiency. Generally, a higher charge rate is more efficient. Keep in mind that NiMH batteries don't take kindly to unnecessary overcharge.

When using constant-current charging, a current is continuously applied throughout the charging phase. Charge acceptance-which is charge efficiency-is enhanced by charging at rates as high as the particular battery type will allow. Of course, this high rate must be cut back as soon as a full charge is reached. Ultimately, continuous charging causes polarization in the electrolyte, which increases battery resistance. With this increase in resistance, a higher voltage is required for a given charge rate, and this in turn contributes to cell heating. Intentionally discharging a NiCd battery to varying depths of discharge prior to charging helps to inhibit the voltage depression effect (better known as memory effect). Discharge before charge most frequently involves taking the battery down to its end-of-discharge voltage which, most of the time, is immediately followed by the initiation of the charge cycle.

Pulsed charging, where a slug of current is followed by a rest period, provides more efficient charge reactions than continuous constant-current charging. For example, using a 2-second interval, you could hit a 1-AH battery capable of withstanding a 1C charge rate with a 2-amp current for 1 second followed by 1 second of rest time. The effective charge rate, therefore, would amount to 1C. The heavier current delivery could, in this case, improve the charge efficiency by up to ten percent over continuous charging.

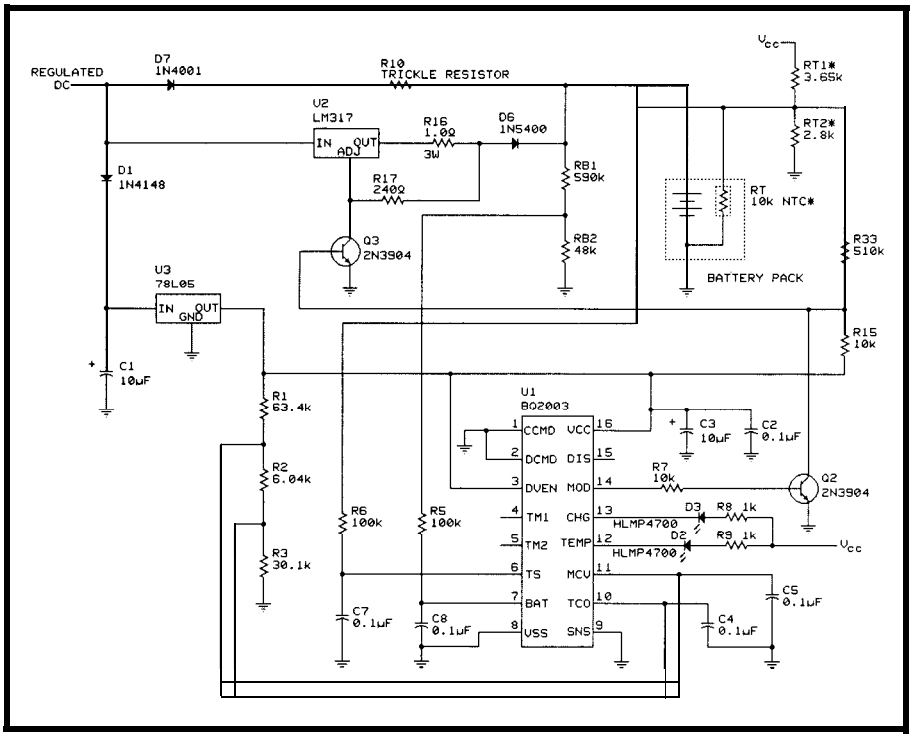


Figure 3—*The bq2003* circuit can *drive an LM317* to *provide maximum charging currents of up to 1.5 amps,*

The rest period allows for cooling and for passive electrolyte depolarization. Depending on how much current you pulse into the battery, the end result could be a similar temperature gain to that of continuous charging but with a shorter charge time. With lower current pulses, less temperature gain would occur with a similar charge time as with continuous charging.

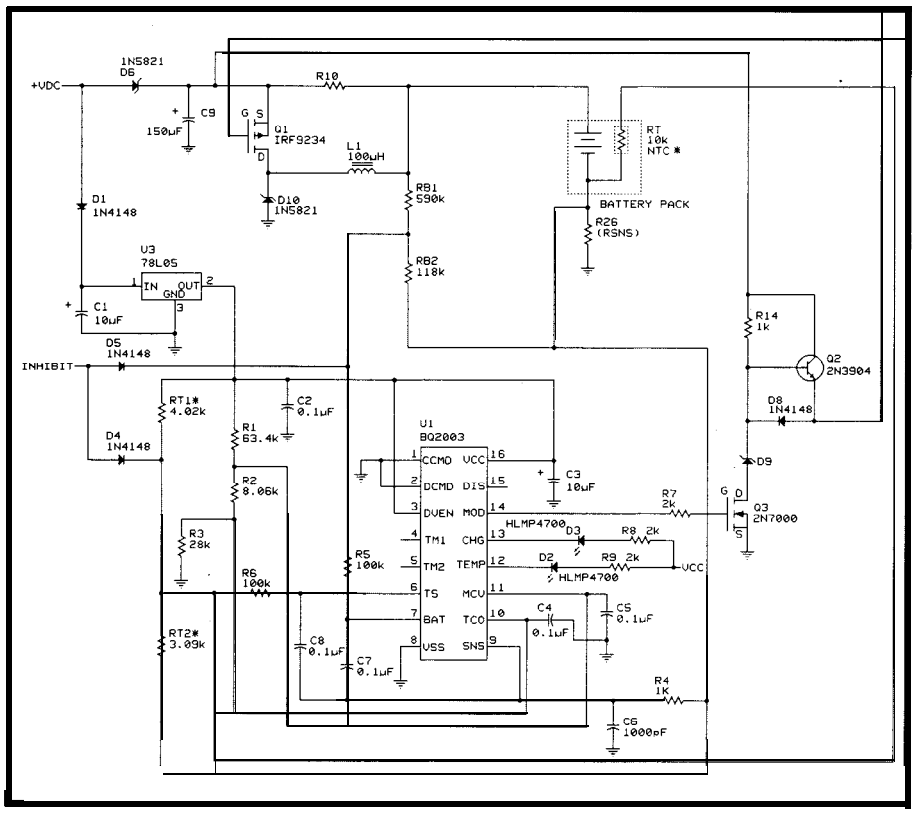


Figure 4-A complete charging system using a p-channel *switching transistor can charge 4–12 NiCd or NiMH cells at currents up to 3 amps.*

Modifying this approach to include a brief discharge period results in a method known as *burp charging.* Here, the 2-second interval could be arranged with a 1.05second charging pulse at 2 amps, followed by a 0.005 second 2-amp discharge, followed by an optional 0.9-second rest period. The brief discharge actively depolarizes the electrolyte, which tends to keep it in a low resistance state. Charge efficiency may increase another five percent above that attained by using standard pulse charging.

By reducing the on-time to a very short interval, the effective current delivery can be choked back to a very low trickle level. Although not really advantageous from a battery conditioning standpoint, this method does provide an easy way to adjust the current flow without resorting to trimming component values. Figure 6 shows how these variations look.

IT'S A GAS

Batteries provide juice to electrical circuits just like your gas tank provides juice to your car's engine, thus a capacity determination methodology is defined and a new term is coined. There is some merit to this analogue, but there are some problems in taking this terminology too literally. From the user's perspective, this idea works well. Easily grasped, especially when depicted graphically, it's handy to be able to determine the amount of fuel that is available to power your circuit. It turns out that the problems associated with gas gauging are of a technical nature and stem from the fact that the size of the "gas tank" itself varies.

First of all, the C rate designation defines the minimum capacity under nominal conditions. In addition to this ambiguity, the capacity will vary throughout the course of the battery's cycle life. It's not unusual for NiCds to start out at 80% of their rated capacity, not acquiring 100% until being cycled several times, finally falling back to 80% at the end of the life cycle. With this degree of variation, any attempt at capacity estimation can prove to be disappointing.

Accurately determining a battery's available charge at any given time can

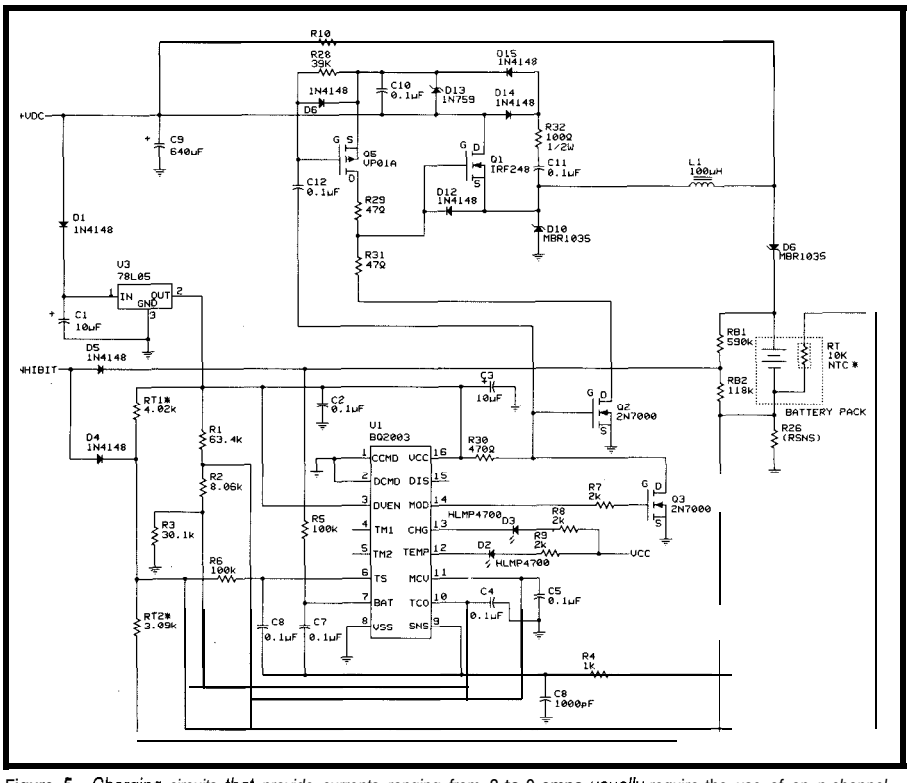


Figure 5—*Charging circuits that provide currents ranging from* 3 to 9 amps *usually* require the *use of an n-channel* power *stage.*

be a difficult proposition. The first thing you need in order to arrive at the battery capacity is the full-to-empty value. Full charge is the point of cutoff at which the charger terminates fast charge. The empty level is the end of discharge voltage that is used as the reference for shutting down the system to prevent damaging depletion of the battery. With NiCd and NiMH batteries, the voltage drops like a rock as it approaches the end-of-discharge voltage, whereas lead acid batteries approach this threshold more gradually. In both cases, however, the levels are clearly defined. You also should factor self discharge into the equation if the system is to remain idle for periods of time. Self discharge usually amounts to 1% per day for NiCd batteries and about 2% per day for NiMH types, but these levels do vary over temperature. Charge acceptance can also undergo a great deal of change, depending on cell type and charge rate and temperature.

Finally, as if that's not enough, realize that the charge cycle may be terminated prematurely resulting in a partial recharge. Even if you manage to get it right, you might have to contend with someone replacing the battery and blowing the whole deal. For this reason, if the battery is not captive to the electrical instrument, the capacity monitoring circuitry really should be included as part of the battery pack.

The idea behind gas gauging a battery centers on the ability to monitor the incoming and outgoing currents by using a small-value sense resistor in series with one of the battery leads. The absolute battery voltage must also be monitored in order to determine the point at which the battery goes empty. Essentially, the method integrates current over time and meters the charge using a counter. When the battery discharges through the sense resistor, the resulting voltage is monitored by the gas gauge circuitry and a counter is clocked in accordance with the current drain. Likewise, during charging, the current/time product is integrated as charge is added to the battery. This would seem to indicate that this measurement method could be used to determine the end-of-charge point accurately for purposes of charge termination. Although this is true, if the gas gauge register somehow lost synchronization with the battery, serious problems could ensue. Because

of this risk, most fast-charge ICs don't make use of this capability as a means of charge termination.

Refer to the manufacturer's data sheets to get the general idea behind the capacity gauging algorithm used by the bq2010 gas gauge IC. Since you won't be able to get silicon for several months, I won't go into any further details on this IC at this time.

ALL BUT THE KITCHEN SINK

Now, I'll briefly touch on a battery management IC that is so complex that I won't even attempt to give you more than an overview of its functional capabilities. This fiendishly complex peripheral is called the bq2001 energy management unit—EMU for short. Not only incorporating a complete fast charging system that includes continuous charging, pulse charging, burp charging, and programmable pulsed trickle charging functionality (along with the usual primary and fail safe charge termination options), the bq2001 also contains a gas gauge, nonvolatile lithium-based RAM (using an external cell], and a serial microprocessor interface. The bq2001 is EEPROM based and allows the programming of default operational parameters in order to allow it to operate in a stand-alone fashion without the need for any processor intervention. Also included on-chip are a bunch of dedicated and programmable output pins that can be used for control, status, or as general-purpose outputs along with a backup cell output for powering external nonvolatile RAMs, RTCs, or other low-level loads. A built-in charge pump serves as a voltage doubler that allows the use of an n-channel FET as the charge control power switch. A control function is provided to drive a p-channel FET that would be used to control battery power to the system.

That's what it has. How it works is another matter entirely; frankly this thing is a challenge to understand. What's most amazing is that this part is now a few years old! Keep in mind that when it was originally developed there was nothing quite like it...and there still isn't. This thing is destined to become a classic.

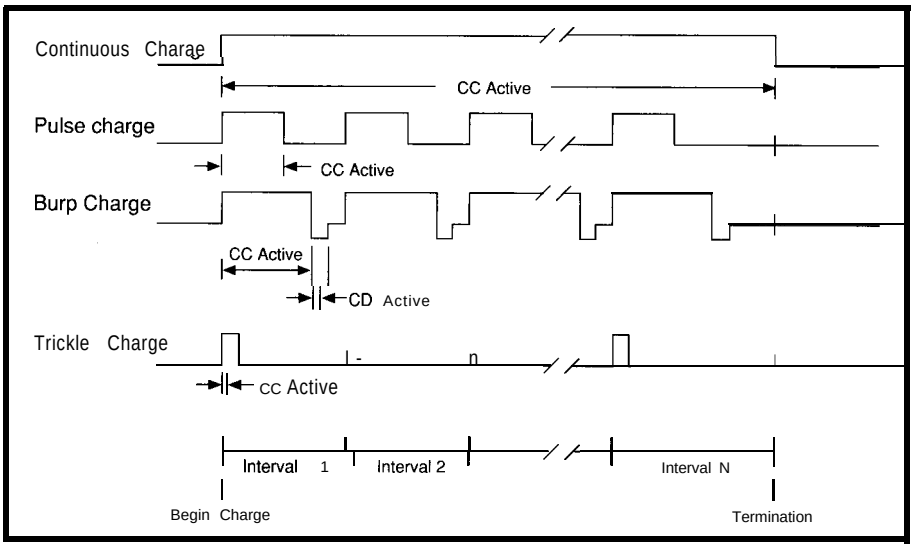


Figure 6—*There are more ways to charge a battery than simply applying a constant voltage or current to if. A dedicated battery management IC can handle all manner of charging methods. Which you choose depends on your battery and a&cation.*

Obviously, it is intended for high-end computer products that can tolerate the significant engineering effort required to put it on the air; this isn't the kind of thing you're going to get fired up in your basement some spare evening. However, you could get your feet wet gradually by putting it on-line a piece at a time. This would be a good idea with such a complex peripheral. For example, if you needed a multistage burp charger, it wouldn't be too difficult to program the relevant EEPROM register to obtain this functionality.

While many ICs come with evaluation boards that you can use to test-drive the circuits, the bq2001 features a full-blown development system. The development system is centered around an 80C32 controller with an on-board 10-bit A/D converter (for battery characterization), 32K of nonvolatile RAM (for storing historical battery data), an RS-232 interface, a bunch of indicator LEDs and test points, and an EEPROM programmer that includes zero insertion force sockets for DIP and SOIC versions of the EMU IC. Beyond its use for application development, the development system lets you get comfortable with a somewhat intimidating chip. Using a natural language interface you can realistically exercise all of the bq2001's capabilities, collect and analyze accumulated data, and unravel the chip's inner mysteries. Photo 1 shows what the development system looks like.

US AGAINST THEM...

So now that I've given you a cursory overview of the battery management arsenal at our disposal, we should be pleased because of the mighty armament we possess. Surely we can gain the upper hand in the quest for battery supremacy. But wait. They can get it too! The playing field is again level. The only clear winners are the end users and the guys who came up with this magic. ❑

I would like to express my thanks to Benchmarq's Mike Calise for supplying information and materials for this article.

John Dybowski is an engineer involved in the design and manufacture of hardware and software for industrial data collection and communications equipment.

SOURCE

Benchmarq Microelectronics, Inc.
2611 West Grove Dr., Ste. 109
Carrollton, TX 75006
(214) 407-0011

I R S

422 Very Useful
423 Moderately Useful
424 Not Useful

PATENT TALK by Russ Reiss

Recent dialogue on the Circuit Cellar BBS has centered around the need for designers to be more aware of the special needs of physically impaired users. It occurred to me that the vast majority of designers probably are unaware of both the needs of the handicapped as well as recent developments in electronic devices to aid them. While this column can do little to rectify the former concern, it can bring to light examples of recent developments which are of potential value to the impaired. As I promised in the BBS dialogue, here is a whole column devoted to this important topic. With luck, perhaps it will stimulate some bright designer who will make a worthwhile contribution.

In searching the patent database, I found that patents related to devices for the handicapped seemed to cluster into three general categories: those for the physically impaired, visually impaired, and speech or hearing impaired. As the first three patent abstracts show, sometimes devices intended for the handicapped may also be applicable to many other areas as well. AT&T's "Written Language Parser System" in Abstract 1 promises improved speech synthesis quality from "freely generated text sequences." It goes a step beyond just the synthesis of sounds, by applying heuristic processing of the output in order to enhance intelligibility by translating abbreviations and special terms, correcting misspellings and noise, and changing word emphasis and pauses.

Abstract 2 (which actually covers both patents 4,950,069 and 4,973,149) from the University of Virginia presents a system for detecting eye movement and for determining the direction in which the viewer is looking. Based on an IR LED and IR-sensitive TV camera arrangement, the system homes in on the "bright eye" effect of light reflected off the eye. Special processing of the signal promises to yield rapid response and "highly accurate resolution." As mentioned in the abstract, such a device serves handicapped persons and also has applications in cockpit and industrial settings.

Abstract 3 presents a pneumatically controlled switch interface. It offers the handicapped person a means of interacting with their computer and other electronic devices. Conventional, commercially available software may continue to be used on the computer since the device simulates existing input devices. One might envision other nonhandicapped uses for such a pneumatic input device in explosive or otherwise electrically hazardous areas.

Abstract 4 represents two patents by Adam Jorgensen (4,907,136 and 5,107,467) which present an ultrasonic apparatus for giving the visually impaired user more knowledge of his surroundings and for aiding him in navigation. Using sonar echo principles much like that found on Polaroid cameras, a narrow beam of ultrasound, presumably emanating from the user's cane, may be directed at objects in any direction and provide the user with an indication of distance to the object. As with all human aids, the "man-machine" interface (direction indicator in this case) is crucial to the success of the concept. A review of the complete patent should provide specifics in this area.

The patent described in Abstract 5 promises to aid the travel of blind individuals through the use of a radio-frequency message apparatus. Basically, a low-power portable radio transceiver carried by the person is employed to query any number of distributed base transceivers. The base unit within range responds by sending a "canned

1

Patent Number	5,157,759
Issue Date	1992 10 20
Inventor(s)	Bachenko, Joan C.
Assignee	AT&T Bell Laboratories
US References	3,704,345 4,278,838 4,674,065 4,831,654 4,868,750 4,872,202 4,873,634 4,975,957 4,996,707 5,060,154
Title	Written language parser system
Abstract	An enhanced text-to-speech synthesizer accepts freely generated text sequences of words and synthesizes the received sequences with proper emphasis and with properly placed pauses. In combination with other elements, the synthesizer provides for an enhanced Dual Party Relay Service where the text generated by the sound-impaired party is synthesized without an attendant's intervention. The text generated by users is made more intelligible by interpreting abbreviations, correcting errors (misspellings and "noise"), translating special terms that are used by the community of users, deemphasizing words based on syntactic considerations and inserting pauses to enhance intelligibility.

PATENT TALK

Patent Number	4,950,069
Issue Date	19900821
Inventor(s)	Hutchinson, Thomas E.
Assignee	University of Virginia
US References	3,986,030 4,623,230 4,648,052 4,836,670
Title	Eye movement detector with improved calibration and speed
Abstract	A system for eye movement detection is disclosed that utilizes an infrared light emitting diode mounted coaxially in front of the lens of an infrared sensitive video camera for remotely making images of the eye of a computer operator. The reflected light causes bright eye effect which outlines the pupil as brighter than the rest of the eye and also causes an even bright small glint from the surface of the cornea. The computer includes graphic processing which takes a video image, digitizes it into a matrix of pixels, and analyzes the matrix. Using special algorithms, the analysis calibrates the system to provide a highly accurate resolution and has a quick scan technique to rapidly determine the location of the pupil's center and the location of the glint relative to each other and with this information determines where the eye is gazing. If the eye-gaze is for a predetermined time at images in selected areas on the computer screen, the area is selected and results in actuation of other devices or the presentation of additional images on the screen. This is especially usable for handicapped persons to control their environment. Other uses include operator interfacing with workstations, cockpit controls, and in industrial environments.

Patent Number	5,126,731
Issue Date	19920630
Inventor(s)	Cromer, Jerry E., Jr.
State/Country	SC
US References	3,229,059 3,848,249 3,911,316 4,207,959 4,298,863 4,453,043 4,562,432 4,567,479 4,706,067 4,746,913 4,865, 610 4,871 ,154 4,979,094
Title	Pneumatically controlled, user-operated switch interface
Abstract	A pneumatically controlled, user-operated switch interface which allows a physically disabled person to operate electronic equipment such as a computer, television, video cassette recorder, and a remote control includes apparatus providing at least one airway passage; first switching circuitry for producing a plurality of switching signals and having at least one pneumatic switch responsive to air pressure in at least one airway passage; second switching circuitry settable in first and second switch positions for selectively connecting each of the plurality of switching signals to selected inputs of the electronic equipment as the electrical input signals, and user-activated apparatus for setting the second switching circuitry in the first and second switch positions. The switch interface can operate a plurality of computer input devices to allow a physically handicapped person to use commercially available software packages.

message" which might identify the location of the base unit by street intersection or landmark, for example. It occurred to me that such a unit might naturally be incorporated within a traffic light design. Located there is the needed operating power, an excellent line-of-site RF position, a natural location at intersections, and space to house the transceiver.

Another device, which is actually designed to be located within the traffic light, is the tactile crossing signal indicator of Abstract 6. The purpose of this aid is to let the user know when it is safe to cross a street (at an intersection). While the abstract concentrates more on the tactile indicator design, I could envision the indicator using, instead, the same type of canned-message, audible response as in the foregoing abstract. But it would take a blind user to say which of the two types of response mechanisms would actually be more useful in practice. In general, it is crucial to involve the handicapped user community in the design of a successful aid. It would also seem that there is no real need for two-way communications. A simple, inexpensive, low-power transmitter incorporated within the traffic light could broadcast its street-intersection information along with the traffic-signal status. If this were in the form of audible information, it might be possible for the blind user to employ nothing more than a conventional, broadcast radio tuned to a specific channel. I believe the more that can be done to lessen the need for the handicapped user to purchase special (and typically expensive) equipment, the more successful the concept will be.

The final pair of patents relate to the hearing impaired. The first of these, presented in Abstract 7, provides a visual indication of the direction and strength of sounds emanated

PATENT TALK

Patent Number	5,107,467
Issue Date	1992 04 21
Inventor(s)	Jorgensen, Adam A.; Jorgensen, Otto A.
Assignee	Jorson Enterprises, Inc.
US References	2,500,638 2,580,560 3,366,922 4,292,678 4,712,003 4,761,770 4,907,136
Title	Echo location system for vision-impaired persons
Abstract	Echo locating apparatus for a vision-impaired person which includes: a sound emitter for emitting a stream of sound bursts of ultra high frequency; at least one receive channel having a microphone for receiving echoes of the sound bursts and generating echo signals; an echo profile detector for generating an echo profile signal of each echo signal; a delay circuit for adding variable delay to the echo profile signal, wherein the variable delay increases with the distance to the reflecting at a diminishing rate of increase. The sound burst emitter is preferably arranged to emit a beam of sound bursts having a given beam angle that can be pointed in any direction.

Patent Number	5,144,294
Issue Date	19920901
Inventor(s)	Alonzi, Louis W.; Smith, David C.; Burlak, Gary J.; Mirowski, Marion
Assignee	LDJ Industries, Inc.
US References	2,255,055 3,495,213 3'922,685 3,973,200 4,225,953 4,253,083 4,495,495 4,598,272 4,660,022 4,754,266 4,935,907 4,961,575 4,998,095
Title	Radio frequency message apparatus for aiding ambulatory travel of visually impaired persons
Abstract	A radio frequency message apparatus for aiding ambulatory travel by handicapped persons such as blind individuals The apparatus generally comprises a portable, radio frequency transceiver, and a stationary radio frequency base transceiver unit. The portable radio frequency transceiver is carried on the person of the handicapped individual and transmits a message request signal in response to manual activation of a transmit button thereon by the handicapped individual. The message request signal is received by the base transceiver, which causes the base transceiver to transmit a prerecorded message signal back to the portable transceiver unit in radio frequency form. The message signal contains location identifying information such as the streets of an intersection at which the base unit is located. The portable transceiver has a limited transmission range of preferably about 20-50 feet to enable it to interrogate a single base transceiver unit located at an intersection within a metropolitan area or at a display/exhibit within a recreational facility such as a zoological park without accidentally interrogating base transceivers in the near vicinity of the desired base transceiver. In a preferred embodiment, an electronic compass is included within the portable transceiver to further aid a visually handicapped individual in orientating himself/herself with respect to North, South, East and Westerly directions.

Patent Number	5,103,223
Issue Date	1992 04 07
Inventor(s)	Humphrey, Jerry J
State/Country	CA
US References	494,337 2,461,448 2,754,505 4,139,742 4590,474 4,635,287 4,851,836
Title	Street crossing signal
Abstract	A street crossing signal for the visually impaired is disclosed. The signal acts cooperatively with the traffic signals to provide a tactile indication of the proper time to cross a street. A vibrator unit is retained relative to a panel having an indicator window there, through which allows contact with the vibrator unit. The vibrator unit is retained relative to the panel and frame so that vibration of the unit is not transferred to the panel or the frame.

near the user. When embedded within a pair of eyeglasses, for example, the user would be directed toward the source of sounds. The benefits could range from the convenience of knowing that someone out of the line of sight is speaking, to the safety of knowing about an oncoming vehicle.

Relatively straightforward electronics make up this device, showing that such aids need not be overly complex.

Finally, the "electrotactile vocoder" of Abstract 8 from the University of Melbourne begs further study of the full patent. It appears that the device permits the user to receive

PATENT TALK

Patent Number	5,029,216
Issue Date	1991 07 02
Inventor(s)	Jhabvala, Murzban D.; Lin, Hung C.
Assignee	The United States of America as represented by the Administrator of the National Aeronautics & Space Administration
US References	3,568,144 3,626,365 3,927,388 4,212,085 4,712,244 4,794,394
Title	Visual aid for the hearing impaired
Abstract	A multichannel electronic visual aid device which is able to signal to the user whether sound is coming from the left or right, front or back, or both. For the plurality of channels, which may operate in pairs, the sound is picked up by a respective microphone and amplified and rectified into a DC voltage. The DC voltage is next fed to an analog-to-digital converter and then to a digital encoder. The binary code from the encoder is coupled into a logic circuit where the binary code is decoded to provide a plurality of output levels which are used to drive an indicator which, in turn, provides a visual indication of the sound level received. The binary codes for each pair of channels are also fed into a digital comparator. The output of the comparator is used to enable the logic circuits of the two channels such that if, for example, the signal coming from the right is louder than that coming from the left, the output of the logic unit of the right channel will be enabled and the corresponding indicator activated, indicating the sound source on the right. An indication of the loudness is also provided. One embodiment of the invention may be carried by the hearing impaired or deaf, as a system, for example, which is embedded into eye glasses or a cap. Another embodiment of the invention may be integrated with a vehicle to give a hearing impaired or deaf driver a warning, with a directional indication, that an emergency vehicle is in the vicinity. In this second embodiment, the emergency vehicle transmits an RF signal which would be used as an enabling signal for the visual aid device to avoid false alarms from traffic and other sound sources in the vicinity of the driver's vehicle.

Patent Number	4,982,432
Issue Date	1991 01 01
Inventor(s)	Clark, Graeme M.; Blarney, Peter J.
State/Country	AUX
Assignee	University of Melbourne
US References	2,150,364 3,612,061 3,831,296 4,390,756 4,441,202 4,581,491
Title	Electrotactile vocoder
Abstract	An electrotactile vocoder for persons having impaired hearing in which electrical stimulation is applied to a multiplicity of electrodes in contact with either side of each finger so as to electrically stimulate the digital nerves of the user under the control of stimulator circuitry which is in turn controlled by processing circuitry for a speech signal received by a directional microphone worn on the ear of the user. The speech processor is suitably of the type described in U.S. Pat. No. 4,441,202 Tong et al. modified to cause stimulation of the digital nerves via the eight finger electrodes and a common electrode held in contact with the wrist of the user.

speech patterns via electrotactile stimulation of the fingers. Since the mechanism for comprehending speech is thought to be extremely complex and deeply embedded in the brain, one wonders what sort of speech patterns can be recognized and utilized by the user of such a device. Nevertheless, it is an intriguing concept with many possible applications even short of full speech. ❑

***Russ Reiss holds a Ph.D. in** EE/CS **and has been active in electronics for over 25 years as industry consultant,** designer, college professor, entrepreneur, and company president. Using microprocessors since their inception, he **has incorporated them into scores of custom devices and new products. He may be reached on the Circuit Cellar BBS or on CompuServe as** 70054,1663.*

SOURCE

Patent abstracts appearing in this column are from the Automated Patent Searching (APS) database from:

MicroPatent
25 Science Park
New Haven, CT 06511
(203) 786-5500 or (800) ***648-6787***

IRS

425 Very Useful
426 Moderately Useful
427 Not Useful

CONNECTIME conducted by Ken Davidson

The Circuit Cellar BBS
300/1200/2400/9600/14.4k bps
24 hours/7 days a week
(203) **871-1988—Four** incoming lines
Vernon, Connecticut

This month, we're going to start off with a discussion of relativity and time. What does Einstein have to do with computer applications? Read on to find out.

Next, we look at some simple methods for defecting the zero crossing *of* an AC *signal.*

Finally, we move info automotive data collection and some of the hazards associated with automotive electronics, though with a twist.

It's all a matter of time

Msg#:11519
From: TERRY NORRIS To: ALL USERS

At work we recently took shipment of an HP cesium beam frequency standard. It had an option that took the accuracy of the 10-MHz output to 10^{-12} (I think). The specifications for this device are incredible, but it raises a question.

In "A Brief History of Time," Hawking says that time is relative to a viewer and his gravitational field. He even says an early experiment about two clocks [one at the base of a water tower, and the other at the top) showed that the one closer to a gravitational field ran slower than one farther away. He later says that without this knowledge, we couldn't have satellites because of the time differences between Earth stations and the satellites.

Is it valid to have a superstable NIST traceable cesium beam with such precision, and a possible source of error due to local variations in gravity greater than the precision? What are the errors introduced due to gravity?

I guess I will have to make a small list of things I believe currently that make my question valid: 1) That technology allows my new cesium beam to be more or equally stable as the early experiment. 2) That the difference might be substantial over time (I know; we don't intend to dispose of the cesium beam soon]. 3) That I understand that short-term differences might be too darn small to care about (like when calibrating a good counter).

Msg#:11611
From: DAVID PARRISH To: TERRY NORRIS

I wouldn't worry too much. According to Einstein, time dilation is given by:

$$T^* = T / \mathrm{sqrt}(1 - (V^2/c^2))$$

with c = 2.3 x 10^9 km/s (velocity of light).

In other words, if the velocity difference is 10,000 MPH, the difference in the times is in the twelfth decimal place!

Msg#:11630
From: DAVE TWEED To: TERRY NORRIS

Sure it's valid to have a clock accurate enough to measure relativistic effects. That's part of the fun of owning one (or two :-)). One experiment that has been performed was to put a clock on each of two jets, one of which circumnavigated the globe over the equator flying east, the other did the same thing flying west. Relativity predicts that one of the planes will see slightly less centripetal acceleration (earth's rotation-air speed vs. earth's rotation + air speed) and therefore a slightly faster passage of time. Sure enough, the two clocks disagreed by the amount predicted by the theory when the planes met again. I don't remember, but I think the difference was on the order of 10^{-9} second.

Also, you can't really call these effects "errors." The clock is accurately measuring the passage of time; it just isn't necessarily the same amount of time as at Ft. Collins. If you want to know what time it is in Ft. Collins, call up NIST.

I read "A Brief History of Time," but I don't recall the comment about satellites. I don't agree with the "you can't have satellites" comment-the frequency errors introduced by relativity are many orders of magnitude smaller than the Doppler shifts caused by the motion of the satellite in relation to the Earth (even geosynchronous satellites move around). Ground equipment is designed to handle this.

Msg#:11936
From: TERRY NORRIS To: DAVE TWEED

I think I finally found something that talks of my question. But first an explanation, or shall I say apology, Hawking didn't say it was impossible for satellites; he just said the differences could cause calculations of positions to be miles off (fifth paragraph from end of chapter 2). Another: You are most decidedly correct when you said the correct term should be differences.

Anyway, the answer was in an astrophysics book. The explanation is simple, but the equation is even simpler:

$$dt = SQRT(1 - 2MG / rc^2) dto$$

dto is the interval between ticks of a standard clock as measured by a distant observer; M = mass; G = the universal gravitational constant, r = radius distance; and c = the speed of light.

The book even says, "Experiments comparing Earth-based and airborne clocks have shown that the gravitational time dilation described by (the equation) occurs. In a series of 15-hour flights at 30,000 ft., the time dilation was 47.1 x 10^{-9} seconds."

I find that very fascinating, so I am indeed able to notice a difference in time due to gravity with my cesium beam. I wonder what the difference is due to me at sea level and NIST! Probably small; I'll figure it out later. I thought I'd write this first.

Msg#:11985

From: BOB PADDOCK To: TERRY NORRIS

I can't let a good discussion on time travel by me without comment, especially if I can get in some relativity bashing along the way.. :-)

Let's start with the conventional. This part taken from "A Matter of Time," by Richard S. Moseson N2BFG, CQ magazine, December 1985, pages 35-38.

"How do atomic clocks work? And who invented them? As Roger Beehler of the National Bureau of Standards explains the clock's operation, cesium atoms are put into a tube called a ***resonant cavity,*** inside a long beam machine which is the atomic clock. The atoms are irradiated with an electromagnetic field and they align themselves in the field with one magnet. They flip back and forth at a fixed rate, and keep doing so as long as the field is at the exact resonant frequency. (If the field is off frequency, the atoms do nothing.) That rate, when the atoms are flipping, is exactly 9,192,631,770 per second. Conveniently, the frequency needed to make them flip is 9.19263 1770 GHz, and the count of the flipping atoms is fed back as a frequency standard to keep the field on frequency [phase-locked loop, PLL].

"According to Dr. Winkler of the Naval Observatory, the idea of an atomic clock was first suggested 40 years ago in a lecture by Professor A. Rabi of Columbia University. The oscillation of the cesium atom was first observed in 1952 by Harold Lyons of NBS, according to Beehler. The first atomic standard in full-time operation was at Britain's National Physical Laboratory in 1955."

Now for the fun "anomalies." If I remember my conventional physics correctly, the charge of an object should not affect its mass or its moment of inertia (time). But we have, from "An Electrically Charged Torque Pendulum," by Dr. Erwin J. Saxl, Pin Hill, Harvard, Mass., "Nature," vol 203, pp 136-138, 7/11/64:

"Unexpected phenomena were noted as follows: (1) When the pendulum was charged electrically with different, carefully controlled electrostatic voltages (together with its equipotential shields), it was observed that positive and negative charges caused different delays. A positive charge caused the pendulum to rotate slower, as a rule, than when the pendulum was charged negatively. The grounded pendulum swung fastest (there are exceptions to this rule at times]."

This is supportive of the life-long work of T.T. Brown, who also showed that mass, in relation to space, could be affected by electric potentials.

Next refer to "The Possibility of the Experimental Study of the Properties of Time" by N. Kozyrev, JPRS: 45238, 2 May 1968 (the document is available from the National Technical Information Service [NTIS], an agency of the U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161, [703]487-4650 for $9.95 + $3 shipping).

Some have said that what Kozyrev was calling time actually was aether, by another time. His experiments showed anomalies in time. And a odd drift of about 420 km/s which leads us to the next stage: the Silvertooth experiment.

Concerning the Silvertooth experiment: The Michelson-Morley experiment, which did not show any translational motion through an aether or other medium of propagation, was later shown to have a fundamental flaw: The standing waves that are reflected back onto a mirror become phase locked on the mirror, and hence to its motion through space. Silvertooth built a standing wave experiment that avoids the phase locking encountered in the Michelson-Morley setup. It uses a configuration similar to the Sagnac experiment, which many years ago did detect motion relative to an aether. Silvertooth's addition was a sensor capable of measuring the spacing between standing wave nodes.

This spacing is dependent upon the orientation of the apparatus relative to the Earth's motion, and this fact made the Earth's motion measurable. Silvertooth measured the 378-km/s motion of the Earth in this experiment.

Some references are: Silvertooth, E.W., "Experimental Detection of the Ether," Speculations in Science and Technology, vol.10, no.1, page 3 (1987). In that same issue, beginning on page 9, is an excellent "plain English" summary by H. Aspden entitled "On the Silvertooth Experiment" by Erol Torun [7/20/92] from the KeelyNet BBS. [We are heading toward the constellation Leo.]

CONNECTIME

Someone always says that relativity disproved the aether of old. Maybe it did disprove the *static* aether of old. Today's aether is considered a *dynamic* sea of energy in the flux of the vacuum. "Sea In Which The Earth Floats," "Dirac Sea," "Neutrino Sea," and so forth, of quantum physics.

I'll leave you with this: The following statement would be considered heresy by the relativity crowd, but take note of who said it:

"According to the general theory of relativity, space without aether is unthinkable; for in such a space there not only would be no propagation of light, but also no possibility of existence for standards of space and time [measuring rods and clocks] nor therefore any space-time intervals in the physical sense."

-Albert Einstein, from an obscure speech in Leiden, Germany; 1920. [Did you catch the name there: Albert Einstein?]

See "Sidelights of Relativity" by Albert Einstein, London, 1922 p. 23.

Zero-crossing detection

Msg#:11433
From: GREG PRICE To: ALL USERS

Any ideas out there on a simple circuit to detect the zero-crossing point of the AC signal so I can switch a nonlatching relay on and keep it on until a control signal (8255 buffered) goes low. Thought about an AND gate with a flip-flop or a PAL. I am sure this has been done many times and many ways. Any help would be appreciated.

Msg#:11877
From: JOHN CONDE To: GREG PRICE

Well, the easiest way I know of to detect the zero cross would be with a zero-crossing detector chip (3059, 3079 if memory serves). This will output a pulse when the signal crosses zero (in either direction). The pulse can gate an SCR which will keep your relay on. Turning it off is another problem. The simplest would be to have the control signal supply the current to the relay (through the SCR), then, when the signal went low, the SCR would turn off and the relay would drop out. Of course, if the control signal can't supply enough current, you can have it control a transistor that will. Hope this helps.

Msg#:11889
From: PELLERVO KASKINEN To: GREG PRICE

First, get a zero-crossing signal as a narrow pulse. Then feed it into the clock input of a D-latch or make one out of a pair of 3-input NAND gates. One input on each for data, a second one for the cross connection, and the third one tied together for the clock.

I used an optocoupler to provide the basic zero detection, or actually the signal polarity detection. I fed the output to an XOR, directly to one input and through a 0.1-ms RC time constant filter to the other input. Got 0.1-ms narrow pulses on every zero crossing.

Msg#:11915
From: LARRY G NELSON SR To: GREG PRICE

How about a Motorola MOC3011 or similar? These are optoisolators with zero-crossing detect and triac output. Not sure the exact part number you would want, but this could be the ticket for what you are looking for.

Automotive EMI can be a drag

Msg#:15175
From: PAUL CONLIN To: ALL USERS

I am looking for some assistance on an embedded microprocessor data logging system. I think electromagnetic interference is causing havoc with the micro.

The system is a handmade prototype of an on-board automotive data logging system based on the Motorola 68HC11. The environment is particularly brutal: vibration, extreme acceleration, and what may be extensive interference from the engine's ignition system.

The target application is a very high output supercharged drag race car using a magneto-based ignition with mechanical distributor. The high boost pressures present very high cylinder pressures. The voltage requirements to ionize such a spark plug gap must be very high.

The data logger consists of a HC11 E2, 1M DRAM, RS-232 level buffer, signal conditioning op-amps/RC networks, and assorted 74HC*xx* support logic. Basic micro circuits are on a printed circuit board, the memory and conditioning circuits are all point-to-point wired. Board is enclosed in a plastic case. All external wires are shielded, drain wires connected to digital ground (no chassis ground). Separate battery from other electronics on car. Micro is 6 feet from magneto, 3 or 4 feet from closest spark plug wire.

The entire system works on the test bench. It works in the pits with the engine running. After initial teething problems were debugged, all appears to function as designed, but the system will *not* work during a drag run. On return to the PC-based retrieval system in the pits, the on-board buffers are empty as if micro has restarted and/or reinitialized. During one test, the "I'm alive" blinking LED controlled from the RTI subsystem had stopped flashing,

CONNECTIME

only to "restart" later!

To eliminate the loose wire possibilities, I am starting to build a two-sided circuit board. It will have an extensive ground bus/plane with filtering caps everywhere possible and a grounded aluminum enclosure. If anyone out there has some suggestions, I would greatly appreciate them.

Msg#:15187
From: MICHAEL SWARTZENDRUBER To: PAUL CONLIN

How do you keep the parts from getting shook right out of their sockets? Don't those railers rattle the brains of everyone (everything) sitting in them?

Msg#:15218
From: PAUL CONLIN To: MICHAEL SWARTZENDRUBER

The plastic case may be causing my problem. I was under the impression that shielding all wires and placing the system six (or so) feet from the magneto could be enough. I am going to place the board in a grounded case. However, this is not a vehicle that can be started unless it is at the drag strip. With a limited number of passes per season, I would like some suggestions on how to ground the system. Should I have a separate electrical ground and then ground all shields and the case to the chassis? Should the electrical ground be connected directly to the chassis or just capacitivly coupled?

To answer your question, all ICs are "tacked" into their respective sockets to prevent everything from shaking apart. The entire black box is also soft mounted. Thanks for your input.

Msg#:15204
From: ALAN COOK To: PAUL CONLIN

First move MUST be to put it all on a proper PCB. NO sockets. Largest ground planes you can afford. Plenty of suppression on the supply lines. I would decouple power and ground for the CPU and memory (each memory chip) using a balun, 22-µF cap, and 0.01-µF cap in parallel. You might even want to try this across the ground for any output drivers. Don't mount any caps vertically, and epoxy large components in place where possible. If you allow for these components on the PCB design, you can always omit those that prove unnecessary. If you need to keep the case light, at least have a thin mesh screen (grounded] inside the

plastic. Suppress every external input, and isolate any output drivers by using open-collector transistor arrangements or a dedicated line driver. The ULN2803 works well on standard vehicles, but may not prove suitable if you have any high-power drive requirements. Make the PCB as small as is feasible, to minimize wire runs. Make sure any pull-ups/downs are adequate, especially on interrupt pins. In such a hostile environment, I would even be inclined towards surface mount (I usually am :->) as this gives a significant decrease in size and contact resistance.

Msg#:15229
From: JIM WHITE To: PAUL CONLIN

I empathize with the difficulty of your task. I developed the TachTale system and had my share of grief confronting the dirty world of racing with electronics. I never tried putting TachTale on a drag racer (Kenny Bernstein's operation is just down the street, and they seemed to have a pretty strong market position).

I have had some experience with high-impulse ignitions (but not the dreaded magneto), and there is good news and bad news. The bad news is that it is not practical to keep out *all* the induced noise. The good news is that you *may* be able to keep the noise low enough to operate.

A working system will almost certainly have to attack the problem from both sides. Keep out as much of the EM1 as possible. This includes maximum feasible shielding. Consider Numetal or other materials which provide magnetic as well as electric attenuation. Certainly shield the electronics; a plastic case a few feet from a magneto and ignition wires is bound to be less than optimal. Remember that the sensor wires are *terrific* antennas and their shielding is not perfect either.

The other half of the equation is to make your electronic design as noise tolerant as possible. Some of the relevant techniques include the use of all CMOS logic, which is more noise tolerant than TTL and NMOS. Use the highest allowable working voltage to increase the noise margins. Minimize the amount of logic, the number of chips, and the lengths of the interconnects. Heavy power and ground planes improve noise immunity by improving common mode noise rejection.

Watch out for devices that may be especially susceptible to EMI/RFI problems. I struggled with the TI TL7705 (as best I recall the part number]. It is a 5-volt supply monitor and reset generator. I had perfectly good working prototypes (wire wrapped) which did *not* use the TL7705, whose design I then changed in what seemed to be a fairly innocuous manner. The TL7705 was added to the "production" design and put to PCB without prototyping (the prototypes went from my screen to the road with hardly a hitch). Turns out that the internal voltage reference circuit design (which is used in many different TI chips) is particularly sensitive to RFI at around 500 MHz. I didn't learn the exact nature of this problem until it turned up in a different product with a different chip that was failing when a handheld radio was keyed to transmit nearby. Naturally, the failure only occurs when the engine was running at high power, when EMF emissions are at their maximum. The symptom is a unit that resets itself more or less often while operating.

The biggest problem I had in terms of the harshness of the environment were the Formula Atlantic cars with the Ford Cosworth motors. These turned out to have some of the most severe vibration problems around (cars both larger and smaller had less intense vibration). This manifested itself in the failure of the clock crystals over time. Once again, the failure would only occur on-track at maximum stress, sometimes the box would "get lost," but usually would find its reset point and look like a reset while operating. In the pit, the hairline fracture of the crystal did not prevent it from operating. There are two basic types of construction for HC49U-style crystals. One type (the bad kind for us) uses solid flat-ended leads with a slot which the crystal wafer slips into. The other type (the good kind) uses tiny looped springs to hold the crystal. I ended up buying crystals made-to-order, which is not terribly more expensive than off-the-shelf, even in modest quantities. The use of smaller crystals, which are more readily available these days, is clearly a benefit.

We invite you call the Circuit Cellar BBS and exchange messages and files with other Circuit Cellar readers. It is available 24 hours a day and may be reached at (203) 871-1988. Set your modem for 8 data bits, 1 stop bit, no parity, and 300, 1200, 2400, 9600, or 14.4k bps.

ARTICLE SOFTWARE

Software for the articles in this and past issues of *The Computer Applications Journal* may be downloaded from the Circuit Cellar BBS free of charge. For those unable to download files, the software is also available on one 360K IBM PC-format disk for only $12.

To order Software on Disk, send check or money order to: The Computer Applications Journal, Software On Disk, P.O. Box 772, Vernon, CT 06066, or use your VISA or Mastercard and call (203) 875-2199. Be sure to specify the issue number of each disk you order. Please add $3 for shipping outside the U.S.

I R S

428 Very Useful 429 Moderately Useful 430 Not Useful

STEVE'S OWN INK

Engineer, Design Thyself

If the physician is admonished to heal himself, perhaps the engineer's equivalent would be to design himself. During this period, when "downsizing" to regain margins is dangerously popular, many senior or mid-level engineers are being asked to find something else to do, and someone else to do it for.

While I would be the last person to minimize the plight of my brethren who suddenly find themselves in this challenging condition, I will not hesitate to admonish you to design yourselves. Apply yourself to discovering a solution. That's what your professors hoped you would get out of all of the incredibly challenging assignments.

How many of us accepted the siren call of those tempting us with the keys to the palace, and were transformed into paper-pushing, report-writing, mostly managing, desk pilots. Don't get me wrong; I'm sure these tasks serve some useful purpose, but any bean counter can do that kind of thing. So let them do it! Recharge the engineer that still lives and breathes in you. Face it, maybe the need for that kind of engineer is passing, and it is time for a new kind of engineer to rise from the ashes like a phoenix. Perhaps we are experiencing serendipity on a societal scale.

I once heard that one of the biggest reasons for the lack of innovation was that we all got just a little too comfortable. After all, if necessity is the mother of invention, who is going to be willing to bear the pain of labor if we are so complacently numbed that we don't perceive the need to do anything. The proponents of this idea would say that while we slept, those more desperate groups forged ahead because they did not suffer from our "plight."

Where is that creative spark and ingenious curiosity that caused you to struggle to become an engineer? The reason I ask is we really need you now. Never before have we needed innovators to take charge of the slumbering human spirit. Never before have so many young persons needed some direction, some inspiration to prevent their talents from being wasted as a hash slinger. Never before was society in need of a grand reemergence of the entrepreneurial spirit. And maybe, just maybe, you are the one to do it! Hey, every little bit helps.

Look around your community. Is there a group of young persons that could be inspired to care about science and math? If so, take them under your wing by forming an electronics club. Is there a school system that could use some expertise in their computer science or voc-tech programs? Share the wealth of your experience with them, the next generation will benefit from your concern. Is there anything you ever thought would be a neat product, or some service you could perform? Now is the time to do it! You may never have this kind of opportunity again!

So I guess what I'm saying is to ignore the doomsayers. Now is one of the most challenging times in human history. There are a multitude of problems out there that need skilled persons like you. Remember that of all the most remarkable discoveries made by humanity, most were made by individuals, not huge conglomerations. Don't hesitate to be a lone reed in the wind!

Engineer, design thyself, and let the world benefit from having known you.